REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
30-06-2006	final	Apr 2005 - May 2006

4. TITLE AND SUBTITLE
The TextLearner System: Reading Learning Comprehension Final Report

5a. CONTRACT NUMBER
HR0011-05-C-0075

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
Program Code Number 5520

6. AUTHOR(S)
Jon Curtis: Project Manager (primary author)
Michael Witbrock: Principal Investigator
David Baxter, John Cabral, Peter Wagner: Key Contributors
Björn Aldag, Keith Goolsbey, Ben Gottesman, Zelal Güngördü, Robert C. Kahlert, Kevin Knight, Cynthia Matuszek, Dave Schneider, Purvesh Shah, Pace Smith, Brett Summers: Contributors

5d. PROJECT NUMBER
ARPA Order Number U173

5e. TASK NUMBER
000203

5f. WORK UNIT NUMBER
N/A

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Cycorp, Inc.
3721 Executive Center Drive, Ste. 100
Austin, TX 78731

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Defense Advanced Research Projects Agency Information Processing Technology Office
3701 Fairfax Drive
Arlington, VA 22203-1714

10. SPONSOR/MONITOR'S ACRONYM(S)
DARPA/IPTO

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The goal of DARPA's Reading Learning Comprehension seedling was to determine the feasibility of autonomous knowledge acquisition through the analysis of text. This report describes the results of that effort by detailing the capabilities of the TextLearner prototype, a knowledge-acquisition program that represents the culmination of the year-long effort. Built atop the Cyc Knowledge Base and implemented almost entirely in the formal representation language of CycL, TextLearner is an anomaly in the way of Natural Language Understanding programs. The system operates by generating a an information-rich model of its target document, and uses that model to explore learning opportunities. TextLearner uses this model to generate and evaluate hypotheses, not only about the possible contents of the target document, but about how to interpret unfamiliar natural language constructions it encounters. Thus TextLearner is able to do two important types of learning—content extraction and rule acquisition—that establish, the authors would argue, the value of knowledge acquisition from text as a rich and promising area of reasoning-based AI research.

15. SUBJECT TERMS
Natural Language Processing, Disambiguation, Context-Modeling

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE	UU	49	Michael Witbrock
					19b. TELEPHONE NUMBER *(Include area code)* (512) 342-4003

The TextLearner System: Reading Learning Comprehension Final Report

June 2006

Cycorp, Inc.

Jon Curtis: Project Manager (primary author)
Michael Witbrock: Principal Investigator
David Baxter, John Cabral, Peter Wagner: Key Contributors
Björn Aldag, Keith Goolsbey, Ben Gottesman, Zelal Güngördü, Robert C. Kahlert, Kevin Knight, Cynthia Matuszek, Dave Schneider, Purvesh Shah, Pace Smith, Brett Summers: Contributors

This material is based upon work supported by the
Defense Advanced Research Projects Agency
DARPA/IPTO
Reading Learning Comprehension Seedling
APRA Oder No. U173
Program Code No. 5520
Issued by DARPA/CMO under Contract #HR0011-05-C-0075

Abstract

The goal of DARPA's Reading Learning Comprehension seedling was to determine the feasibility of autonomous knowledge acquisition through the analysis of text. This report describes the results of that effort by detailing the capabilities of the TextLearner prototype, a knowledge-acquisition program that represents the culmination of the year-long effort. Built atop the Cyc Knowledge Base and implemented almost entirely in the formal representation language of CycL, TextLearner is an anomaly in the way of Natural Language Understanding programs. The system operates by generating a an information-rich model of its target document, and uses that model to explore learning opportunities. TextLearner uses this model to generate and evaluate hypotheses, not only about the possible contents of the target document, but about how to interpret unfamiliar natural language constructions it encounters. Thus TextLearner is able to do two important types of learning—content extraction and rule acquisition—that establish, the authors would argue, the value of knowledge acquisition from text as a rich and promising area of reasoning-based AI research.

Contents

1 Introduction

This report describes the efforts and results of the Reading Learning Comprehension DARPA/IPTO seedling, contract number HR0011-05-C-0075. The structure of the report is as follows: First, the reader is given a summary of Reading Learning Comprehension's goals, followed by a description of the main technical challenges that had to be met to achieve those goals. The report then describes the methodology behind and the architecture, components, and capabilities of the TextLearner system, the product of this effort. Next, the report describes two experiments designed to test the general utility and promise of a deployed TextLearner, the results of one of which have been described in a recent publication. Following this, the report summarizes the most signification developments from the Reading Learning Comprehension effort. The report ends with a discussion of future research ideas.

2 Task Objective

2.1 Overview

The Reading Learning Comprehension seedling can be viewed as a natural step in the development of knowledge-acquisition methods for large knowledge-based systems. Where the High Performance Knowledge Base (HPKB) project established the value of systems built largely through the manual efforts of trained ontologists, and the subsequent Rapid Knowledge Formation (RKF) project showed that domain experts without a background in ontology could extend a knowledge-base through limited "mixed initiative" dialogue and special-purpose knowledge entry tools, the Reading Learning Comprehension seedling sought to establish the feasibility of an automatic method of knowledge acquisition from and about natural language. The TextLearner prototype, which implements many of the algorithms described in this report, represents a model of learning that distinguishes this effort from most research on learning performed over the past two decades, such as function approximation, reinforcement learning, and Markov model learning, which has focused on statistics-based learning of sequences and patterns.

In describing the TextLearner system, it is important to distinguish the task of building a program that learns *through the analysis of text* from the broader task of building software that can *read*—a problem perhaps as difficult as, if not identical to, the problem of building an Artificial Intelligence. Nevertheless, the approach taken to textual analysis inside TextLearner approximates important features of reading. Foremost of these is the ability to defer resolution of ambiguous words and phrases, and thus "move through" a text, using later information to resolve earlier uncertainties. Towards this end, TextLearner generates an information-rich model of the target text, including a declarative, formal representation of tokenizations, linkages, and parse trees, and stores this representation in a knowledge base primed with thousands of commonsense concepts and rules. This environment—a reified model of context against a backdrop of common

sense—enables TextLearner to generate multiple competing hypotheses about the meanings of words, phrases, and sentences, and to reason about their respective merits. This level of context-modeling also enables TextLearner to approximate another important, if less recognized, feature of reading: the use of context to learn rules that facilitate and improve the process of learning from text. Though it is evident that humans are able to improve both their reading comprehension and writing skills by reading, a precise model of how humans manage this feat was not required to make advances in this area. Rather, familiarity with the core natural language understanding (NLU) capabilities of existing software allowed us to identify small classes of rules that were both tractable to learn and capable of yielding results. The context-modeling and analysis that define the TextLearner system thus enabled us to pursue two important learning objectives simultaneously: *content extraction*, or learning what information a document contains; and *rule acquisition*, the learning of rules that, once added to the system, make it a better learner. Each objective is discussed in turn.

2.2 Content Extraction: The "Reading Learning" Problem

The objective of extracting content from text decomposes into three sub-objectives, each of which is described in its own sub-section:

- **Formalization of Propositional Content**, the objective of producing a record of the propositional content of a document, encoded as sentences of a formal representation language;
- **Semantic annotation**, the objective of producing record of what a document is about, by encoding a (weighted) mapping of string-occurrences into terms of a formal language;
- **Extracted presuppositions**, the objective of producing a formalized record of content, not explicitly declared in a document, but presupposed by word choice, sentence structure and conventions of interpretation

2.2.1 Formalization of Propositional Content

One obvious goal for a natural language understanding technology such as TextLearner is the production of a formal translation of the target text. However, in designing TextLearner, it was determined that this goal was both too unrealistic and (perhaps more importantly) too narrow to serve as the system's sole objective. That this goal is unrealistic is due in part to the inherent difficulty of the problem: despite the fact that many syntactic parsers work reliably, formal languages are typically optimized toward compactness of representation and suitability for inference. As a result, the formal "target" language will often have little in the way of a predictable isomorphism with the structures produced by syntactic parsers. Exacerbating this difficulty was the design decision to do without restrictions in the way of domain, vocabulary, or grammar in the texts to be processed. Restricting any or all of these would increase the prospects of writing a fairly precise semantic translation program; however, as a test of the feasibility of a *general* AI program that learns from and about text, such restrictions would serve to

"game" the outcome in an unhelpful way. After all, in "real world" articles, documents, and conversations—all fair game for sources of text—domains and topics shift rapidly, words are coined and senses change. Success in cases where these elements are fixed and known to the system beforehand thus renders dubious the extensibility of the algorithms employed. The prevailing view among TextLearner's developers was that while a semantic translation would be one target of the system's text analysis, it would not be the only one, nor even the focal one. Of more interest would be whether we could learn rules or facts that would improve or increase the system's ability to produce such interpretations—something that will be discussed below, in section 2.2.

As noted above, focusing on a semantic translation of the target text is not only unrealistic, but also unnecessarily narrow. There are two reasons for this. First, as evidenced by widespread interest in semantic annotation of documents,[1] there is value to be had in gleaning merely what a sentence is *about*, even where this "aboutness" is cast in a quasi-Fregean light, under which a sentence is about every thing referred to by any of its constituents.[2] Notably, textual annotation of this sort can be achieved without understanding the propositional content of a sentence. The second reason to view sentence-by-sentence formalization as too narrow an objective is that sentences convey extra-propositional content—in particular, presuppositions—that will be absent from a formalization of the propositional content. The outputs of a program that extracts content will thus be information poor in certain respects, if it were capable of producing only a sentence-by-sentence formalization of propositional content. Let us examine each of these points more closely.

2.2.2 Semantic Annotation

Above, it was noted that the vocabulary comprising a formal representation language is typically designed so as to maximize inferential utility. Often, this means that a given phrase of natural language might have a "target" translation into a representation language that, while efficient for inference, makes a general algorithm for translating natural language sentences difficult to extract. For example using a special-purpose[3] predicate such as `highestVolcanoInThisArea` to represent "Etna is the tallest volcano

---

[1] Notable among the "hot" topics in this area is the nearly globally embraced "semantic web" movement, which has among its aims the annotation of virtually all electronic documents accessible through the internet. See http://annotation.semanticweb.org/.

[2] See, for example, Frege's "building blocks" analogy on pg. 243 of *Nachgelasene Schriften*, ed. H. Hermes, F Kambartel and F. Kalubach, Hamburg, 1969. Frege's view more naturally pertains, not to natural language categories of constituents, such as *subject* and *predicate*, but to "constituents" as they would be distinguished in a logical notation. So on the view described here, "**F(a)**" is equally about the function denoted by "**F( )**" as it is about the concept denoted by "**a**." The relevance of Frege's view here rests in its similarity to the view of semantic annotation implemented as a term-by-term, phrase-by-phrase mapping into a logical formalism, where each denotation of each term of the formalism thereby stakes an equal claim to being something that the sentence is about.

[3] This predicate actually exists as part of the CycL vocabulary, primarily to serve as an example of both 1) how not to represent knowledge typically, and 2) what one *can* do, if inflexible inferential/representational requirements are imposed (i.e., one's hands aren't tied by virtue of using CycL as the representation language).

in all of Europe"—`(highestVolcanoInThisArea ContinentOfEurope MountEtna-Volcano)`—might make reasoning about Mt Etna's geospatial location relative to Earth's continents very efficient,[4] but it makes for a poor model on which to translate English sentences of the form, *NOUNPHRASE-1 is the tallest NOUN in NOUNPHRASE-2*. It is an even poorer model for *NOUNPHRASE-1 is the SUPERLATIVE NOUN in NOUNPHRASE-2*, and downright useless for *NOUNPHRASE-1 is NOUNPHRASE-2*, despite the fact that the original sentence instantiates all three patterns.

Though the particular example selected here might be more illustrative of the problem than would an example selected at random, the general point nevertheless holds: Without artificial constraints on domain, vocabulary, or grammar, it is highly improbable that one can reliably generate a formalized translation of an arbitrary "legal" natural language sentence by compositional means—i.e., by virtue of having mapped each natural language constituent into a corresponding term in the formalism.[5] However, that does not mean that performing such a mapping is an exercise in futility. Quite the contrary, if such a mapping can be performed automatically, and ambiguities at the NL level can be resolved mostly correctly (again, automatically), the result is a method of automated semantic annotation of documents—a major gain for an NLP community focused on problems such as Information Retrieval (IR—finding a document based on how its content aligns with the entities mentioned in the user's query) and question-answering (traditionally, "fine-grained IR"—retrieving passages, as small as a sentence or phrase, that, due to their semantic properties, appear to answer the user's question).

Given the richness of the ontology in which this semantic representation would be embedded, the prospects for using TextLearner to generate a qualitatively superior brand of annotation appeared promising (and still do). For example, given a document annotation that correctly maps an occurrence of the string "tank" to `Tank-Vehicle` as opposed to `LiquidStorageTank`, one would be able to retrieve the document with search strings that, in addition to "tanks," included non-synonymous, non-hypernymous, and non-hyponymous strings such as "military," "track," "armor," and "battle," given the tight semantic relationships among `Tank-Vehicle` and the terms corresponding to the English words in the target ontology (the Cyc ontology). Given the potential high-

---

[4] Indeed it does. This very sentence is asserted in Cyc. Because of special purpose reasoning modules designed to support transitivity, Cyc knows that to be the tallest volcano in an area is to be geographically subsumed by that area via the relationship (`genlPreds highestVolcanoInArea geographicalSubRegions`). Cyc can thus prove `(geographicalSubRegions ContinentOfEurope MountEtna-Volcano)` without having to transform the problem in a way that triggers other applicable rules.

[5] One can increase one's chances by including "intermediate," or "throw-away" vocabulary in the formalism (such as "referentless" terms for things like determiners and pronouns) that serves to replicate the syntax of the target natural language. Attempts to facilitate NLP by using such vocabulary at Cycorp have met with some success (see Panton, *et al*, "Knowledge Formation and Dialogue using the KRAKEN toolset, in *Proceedings of the Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artifical Intelligence*, pp. 900-905, Edmonton, Canada, July 28-August 1, 2002). The need to resort to this sort of "throwaway" vocabulary is obviated in the TextLearner system by virtue of the richness of the context model it maintains.

payoff and low-risk involved, the scope of the Reading Learning Comprehension objective was extended to move beyond formalized translation and into semantic annotation.

2.2.3 Presupposition

As noted above, a second reason to expand the objectives of TextLearner's content extraction capabilities has to do with the possibility of grasping the "low-hanging" fruit in the area of presupposition. Consider the following pair of two sentence paragraphs:

- "Angola is in Africa. It is a country with a democratic form of government."
- "Angola is in Africa. The country has a democratic form of government."

For each paragraph, a suitable formalization of the first sentence (using mock CycL) might be something as straightforward as,

```
(inRegion Angola ContinentOfAfrica)
```

Translating the second sentence correctly in each case requires a bit more sophistication. Both sentences contain anaphoric expressions—noun phrases whose meanings are determined by the denotations of expressions that occur earlier (their antecedents). Let us suppose a system for translating NL to a formal representation correctly identifies "Angola," and not "Africa," as the antecedent for "It" in the first paragraph. Such a system would then be in a position to correctly assign "It" the same semantics it assigned to "Angola" in the first sentence. We might imagine a target representation of the second sentence to thus be

```
(and
    (isa Angola Country)
    (formOfGovernment Angola Democracy-FormOfGovernment))
```

The conjunction of this sentence with `(inRegion Angola ContinentOfAfrica)` arguably provides a formal representation of the propositional content of the first of the our two paragraphs.

However, when we attempt to translate the second sentence of the second paragraph, things become tricky. Of interest here is the question of what the anaphoric expression "The country" refers back to. Let us suppose the system that handled "It" correctly in the first paragraph correctly identifies "Angola" as the antecedent of "The country" in the second paragraph. As before, this system would be in a position to correctly assign "The country" the same semantics it assigned to "Angola," having established co-reference. This yields a semantic translation,

```
(formOfGovernment Angola Democracy-FormOfGovernment)
```

for the second sentence, which, conjoined with the first, arguably provides a formal translation of the second paragraph. However, there is an important bit of content

provided by both paragraphs, and reflected only in the translation of the first, that is conspicuously absent—namely, that Angola is a country. The fact that Angola is a country is not a claim put forth by either sentence of the second paragraph; rather, conventions for resolving co-reference for "familiar" noun phrases allow one to derive content presupposed in the paragraph as a whole: Angola is, in fact, a country; otherwise the author wouldn't have used "The country" to refer to it.

Like many natural language phenomena, presuppositions are ubiquitous and subtle, and worthy of a dedicated research program. However, given the time and funding constraints of the Reading Learning Comprehension seedling, it was decided that the scope of presupposition-extraction would be limited to "low-hanging fruit"—in particular to cases of anaphora resolution, including pronouns (such as "he," the use of which presuppose the gender of the denotations of their antecedents) and certain classes of appositives (e.g., "the sculptor Michelangelo") that serve to classify an object without conveying that classification declaratively.

2.3 Rule Acquisition: The "Learning Reading" Problem

Though content extraction might be the most recognized and natural objective for systems that implement NLU techniques, an early objective of TextLearner has been that it learn "increasingly autonomously" [6] from text. This means that not only should TextLearner perform some level of content extraction, it should learn in a way that makes it do a better job of learning from text over time. Specifically, this entails finding ways to learn rules that help the system handle new linguistic phenomena.

A handful of high-payoff areas for rule acquisition were identified:

- **Rules for interpreting noun compounds**: Having generated hypotheses about what the head and modifier of an identified noun compound could mean, the system should generate rules explaining the compound—i.e, rules that extend the semantic relationship between the head and modifier semantics (e.g., the semantics of "rain" and "coat") to explain the meaning of the compound.in context (e.g., "rain coat" denotes a type of coat that guards against (as coats are wont to do) rain.
- **Rules for interpreting new verb phrases**. Current mechanisms for translating verb phrases into CycL rely on semantic translation templates. Coverage in this area is substantially incomplete. The system should, when enough information is given in the context model, propose new templates that enable some level of formalization. As will be discussed, productive techniques have been implemented in TextLearner that generate semantic translation templates for verbs that lack templates, as well as extensions for verbs with limited template information when verb phrases with prepositional complements are encountered.

[6] Witbrock, M. and Schneider, D., "Reading – Learning – Comprehension" DARPA proposal, 9/23/2004

- **Rules (templates) to enable Example-Based Machine Translation (EBMT).** EBMT is the process of deriving translation based on examples.[7] Though this technology has been used primarily to translate between natural language sentences, some promise has been shown in the area of applying these techniques to effect NL-to-CycL translations[8], despite obvious issues with respect to the use of natural language devices such as metaphor and pronouns that do not (typically) correspond to elements of a formal representation language.

The details behind each rule acquisition method are covered in the "General Methodology" section below (Section 4).

3 Technical Problems

Though some of the technical challenges that arose in the development of the TextLearner system are common to any software engineering project—e.g., ensuring reliable access to external programs or databases—some of the more interesting challenges are unique to large-scale knowledge-based AI programming. This section describes two such challenges and our responses to them.

3.1 Implementing Procedures in a Declarative System

By design, the bulk of the "code" responsible for guiding TextLearner's behavior is encoded directly into the Cyc KB as CycL rule-assertions that operate in a forward (aggressive), as opposed to backward (when called upon) direction: As information is added to the KB, the various rules that comprise TextLearner knowledge-acquisition components are triggered, resulting in a subsequent round of information being added, and the subsequent triggering of state-changing rules, until a state of quiescence is reached. In such an environment, it is difficult to control the order in which particular rules are triggered, which can be problematic when the addition of information at a later stage is sufficient to invalidate a rule-application that was executed at an earlier stage. This is a version of the age-old problem of elaboration tolerance,[9] which the developers of TextLearner combated mostly successfully by combining non-monotonic features of the Cyc representation language with a more novel mechanism for reconsidering and revising KB content.

3.1.1 Non-monotonicity

Cyc has long had non-monotonic reasoning capabilities, the most mature of which is its ability to store meta-rules about rule applicability in its knowledge base. For example, the Cyc KB contains the following rule:

---

[7] See, for example, http://www-2.cs.cmu.edu/afs/cs.cmu.edu/user/ralf/pub/WWW/ebmt/ebmt.html, Ralf Brown's research page.

[8] See Bertolo, *et al*, Quirk Final Report.

[9] McCarthy, J. "Elaboration Tolerance," *Proceedings of the Fourth Symposium on Logical Formalizations of Common Sense Reasoning*, London, Janary 1998.

```
(not
      (and
          (isa ?ANIMAL Mammal)
          (genls ?ABILITY Flying-FlappingWings)
          (physicallyCapableOf ?ANIMAL ?ABILITY doneBy)))
```

which says that for every mammal *ANIMAL*, it's not the case that there is a kind of flying that *ANIMAL* is physically capable of. Thus if Cyc were to learn of a new object, `Mammal-0135`, for which all was known that

```
(isa Mammal-0135 Mammal)
```

Cyc would be able to prove that `Mammal-0135` cannot use its wings to fly. I.e., it can use the above rule to prove,

```
(not
  (physicallyCapableOf Mammal-0135 Flying-FlappingWings doneBy))
```

However, if Cyc then learns that `Mammal-0135` is not merely a mammal, but a bat,

```
(isa Mammal-0135 Bat-Mammal)
```

we would not want Cyc to be able to prove what it could when it knew only that `Mammal-0135` was a mammal. The rule, as written, still clearly applies; but there is a meta-rule *about* this very rule's conditions of applicability, namely:

```
(exceptWhen
  (isa ?ANIMAL Bat-Mammal)
  (not
      (and
          (isa ?ANIMAL Mammal)
          (genls ?ABILITY Flying-FlappingWings)
          (physicallyCapableOf ?ANIMAL ?ABILITY doneBy))))
```

This rule says that the "mammals can't fly" rule applies to all mammals and all types of self-powered, "wing-driven" flying, except when the mammal in question is a bat, in which case it doesn't apply at all.

The ability to state exceptions in CycL is crucial in handling changes in the knowledge base, particularly where TextLearner is concerned. For example, after the first sentence of a document has been processed, the supporting evidence might weigh in favor of one interpretation of an ambiguous word or phrase. But after the full paragraph has been considered, the evidence might weigh in favor of a rival interpretation. The ability to state exceptions thus helps guard the system against committing to a hypothesis too early.

3.1.2 Selective Truth Maintenance

Unfortunately, the use of exception rules, though necessary to make TextLearner elaboration tolerant, is not sufficient. As noted above, many TextLearner components are

implemented as sets of forward rules, so that their dependents are derived and added to the Knowledge Base as soon as the antecedent conditions of the rule are met. This reintroduces elaboration tolerance all over again for forward inference, which can be illustrated by extending the example, described above, in which `Mammal-0135` is known to be an instance of `Mammal` at time *t*, and only later known to be an instance of `Bat-Mammal` at a later time, *t2*. If, at time *t*, the rule stating that mammals cannot fly were forward, the system would forward derive

```
(not
  (physicallyCapableOf Mammal-0135 Flying-FlappingWings doneBy))
```

at that moment. The later, *t2*, claim, that `(isa Mammal-0135 Bat-Mammal)`, is not sufficient to retract the deduction, as it is merely a claim, like any other, that might or might not be consistent with what the system already takes to be true. If, for example, the KB contains the additional knowledge that all bats can fly, the *t2* claim is, like `(isa Mammal-0135 Computer)`, a claim whose addition to the knowledge base would create an inconsistent state. Indeed, if the inconsistency can be detected immediately by the system,[10] the claim might be rejected altogether.

The general mechanism for handling these sorts of inconsistencies is a Truth Maintenance program, designed to investigate derivations of forward rules, and reconsider their supports. However, this program is typically run only at compile time, and not with the addition of every assertion, which, given the size of the Cyc KB, would be prohibitively expensive.

However, this creates a conundrum for TextLearner, which relies heavily on forward inference and a consistent stream of new data that will trigger forward inference, in many cases prematurely (i.e., before all the information is "in"). The solution decided upon was to implement selective truth maintenance in TextLearner, using a piece of procedural vocabulary—a unary predicate `reconsiderAssertionDeductions`—and forward rules concluding to it. Areas where elaboration tolerance were likely to break down were identified, and then forward rules were added that looked for possible inconsistencies. These rules, when successfully triggered, derived a command to the Truth Maintenance code to investigate and repair as needed.

It would be beneficial to consider an example from one area where selective truth maintenance proved useful to TextLearner. Resolving anaphora, cases of pronouns or definite noun phrases whose meaning depends on the meaning of other, earlier expressions (their antecedents) that these phrases refer-back to.

---

[10] Cyc's agenda control process does some degree of semantic well-formedness-checking, which includes checking to see whether proposed assertions are easily disproved (via, example, simple graph-walking of the type hierarchy). For example, given that Cyc knows in its microtheory (context) of people data (`PeopleDataMt`) that Hu Jintao is a human being (`(isa HuJintao HomoSapiens)`), the agenda control would block the addition of `(isa HuJintao Automobile)` to any part of the KB where the contents of `PeopleDataMt` are accessible.

For example, consider the following rule that serves to identify situations where there is exactly one possible antecedent for an anaphoric expression:

```
(implies
    (and
        (contextuallyPossibleAntecedent ?ANA ?ANTE)
        (extentCardinality
            (TheSetOf ?ANTES
                (contextuallyPossibleAntecedent ?ANA ?ANTES)) 1))
    (onlyPossibleAntecedent ?ANA ?ANTE))
```

This CycL expresses the rule: *ANTE* is the only possible antecedent for the anaphoric expression *ANA*, if 1) *ANTE* is, in fact, a possible antecedent for *ANA* and 2) there is exactly one such possible antecedent. When used backward in inference against a *fully-specified* model of the target text, this rule works as intended: Under such conditions, the number of possible antecedents has been fixed prior to the application of the rule. Unfortunately, in a system such as TextLearner, where rules such as this are run forward and so can be triggered multiple times as the model is constructed, the rule is certain to generate dependents at time `t` that will prove false at some later time `t'`. Indeed, this rule will be satisfied as soon as the first possible antecedent for *ANA* is identified: As TextLearner's analysis of a document proceeds, there will be a time, however brief, where TextLearner has encountered, for the first time, a possible value for *ANTE*, and has yet to discover any others. If others are discovered, the discoveries will come too late to block the deduction: the very first value for *ANTE* has successfully triggered the rule, resulting in the addition of `(onlyPossibleAntecedent ANA ANTE)` as a derived assertion to TextLearner's model of the document.

To combat this, a selective truth maintenance rule was added:

```
(implies
      (and
          (assertionSentence ?ASSERTION
              (onlyPossibleAntecedent ?ANA ?ANTE-1))
          (contextuallyPossibleAntecedent ?ANA ?ANTE-2)
          (different ?ANTE-1 ?ANTE-2))
      (reconsiderAssertionDeductions ?ASSERTION))
```

This rule will be triggered anytime a sentence of the form

```
(contextuallyPossibleAntecedent ANA ANTE)
```

is added to TextLearner's model, guaranteeing that all and only those dervied `onlyPossibleAntencedent` assertions that should survive the document-analysis process do survive.

3.2 Performance

Though forward inference has been utilized heavily in other Cyc-based projects[11] no project has exercised Cyc's forward inference capabilities to the degree that the Reading Learning Comprehension project has. This section describes two areas where this reliance on forward inference posed a significant technical challenge.

3.2.1 Reification Templates

The heavy reliance on forward inference in TextLearner—that is, the use of rules that actively derive conclusions when information meeting their antecedent conditions are added to the knowledge base—combined with the huge amount of data over which reasoning is done, resulted in slow performance. As a fully automated learning program, the performance, or speed of TextLearner would not ordinarily matter. However, early on in the project, slowness became a development obstacle, as it became prohibitively expensive to run regression tests or to test patches.

An important step taken to address the problem was to restrict which rules could fired during TextLearner's text analysis. This was accomplished through the use of templates that specified patterns for restricting the use of rules in forward inference. Prior to the Reading Learning Comprehension project, such templates would specify "legal" forward rules on an explicit rule-by-rule basis. However, given the vast number of forward rules needed to implement TextLearner, a rule-by-rule approach was too unwieldy. A simple solution, a new binary predicate, `creationTemplateAllowsAllRulesFromMt` that could be used to associate a template with a microtheory, allowed all rules asserted in that microtheory to be "legal." Text learner's extensive use of these templates is unprecedented; indeed, it is the only Cyc-based software program that has complete control over how and when forward inference is executed.

3.2.2 Using a "Balanced" Tactician

For many years, Cyc had separate solution-identifying algorithms, known as tacticians, for the inference engine to use depending on whether it was solving a problem during backward or forward inference. The reason for this was primarily one of performance: The more sophisticated, "balanced" tactician used in backward inference worked much more slowly than the simpler, "depth-first" tactician used in forward inference. However, the simpler tactician is prone to error, and as TextLearner increasingly relied on forward inference, these errors emerged ever more frequently, eventually blocking TextLearner from performing certain fundamental tasks.

It thus became necessary for TextLearner to switch to the balanced tactician, initially increasing the time spent in forward inference, already the bulk of TextLearner's total

[11] Most notably in the Vulcan-sponsored Halo project and the commercial MySentient Answers system, described respectively in Friedland, *et al*, "Project Halo: Towards a Digital Aristotle," *AI Magazine* 25(4) pp. 29-48, Winter 2004, and in Curtis, *et al*, "On the Effective Use of Cyc in a Question Answering System," in *KRAQ 2005 Proceedings*, Edinburgh, Scotland: 2005.

time, by 300%. The Cyc inference team then ran several experiments, using TextLearner use-cases to establish baselines, enabling them to identify and repair several causes of slowness. The most dramatic of these repairs involved a new kind of inference heuristic, so-called "preference modules," that facilitate more correct and efficient meta-reasoning about the completeness and productivity of certain classes of problems.

As a result of these efforts, the balanced tactician now operates more efficiently than the depth-first tactician used to. The depth-first tactician has since been discarded, meaning that all users of forward inference in Cyc now benefit from these changes.

4 General Methodology

4.1 The Reading Learning Comprehension Process Overview

At its most general level of description, the TextLearner system executes a three-stage script for analyzing and learning from text. This script is illustrated in Figure 1:

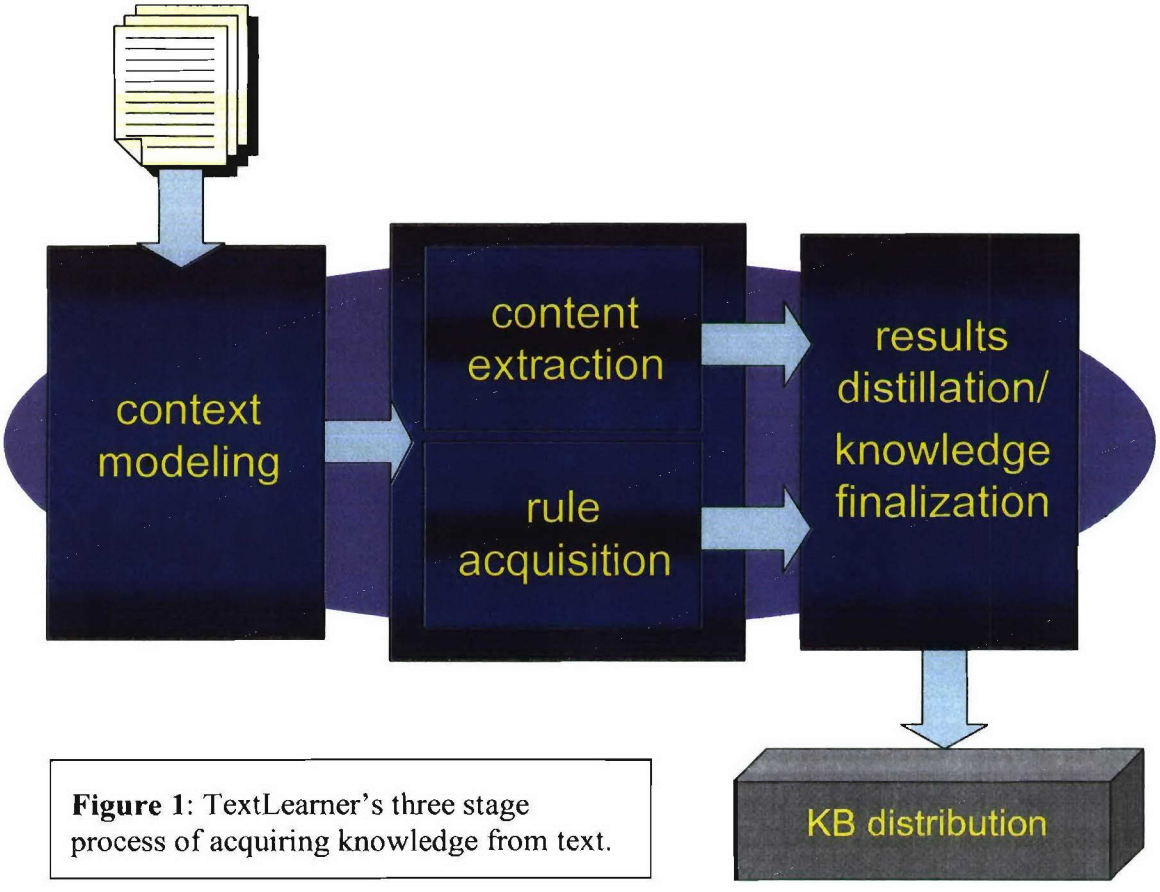

Figure 1: TextLearner's three stage process of acquiring knowledge from text.

Each stage in the process is described in detail below.

4.2 Context Modeling

4.2.1 Representing Document Structure

A number of distinctions in the Cyc ontology are relevant in an explanation of TextLearner's approach to modeling the structure of documents. The most intuitive of these is a distinction between an *authored work*, such as a musical score, a screenplay, or an op-ed piece, and its spatio-temporal *instantiations*—musical broadcasts, theatrical performances, newspaper clippings and the like—by which humans experience the contents of the work. English contains many words that can be used to refer to either a work or a work-instantiation, such as "book," "document," "paper," "article," and "movie." However CycL captures the distinction unambiguously by having distinct terms corresponding to the two kinds of senses. The Cyc term `PropositionalConceptualWork` is meant to denote the class of abstract works that convey propositional (truth-evaluable) content, and so has specializations, such as `Book-CW`, that are useful in representing that the "work" sense of "book" is at play in the sentence "I've read all of Stephen Hawking's books." In contrast, `InformationBearingThing` represents the collection of all objects and events that carry information. Under this collection one finds terms such as `BookCopy`, useful for representing and understanding "instantiation" uses of "book," as in "I lost Stephen's book in the scuffle."

In Cyc's ontology, works and their instantiations are related, definitionally, to the notion of a shared *information structure*, so that something counts as an instantiation of a work if, and only if, it instantiates the structure of that work. This is encoded in CycL as,

```
(implies
     (and
          (instantiationOfWork ?COPY ?WORK)
          (correspondingAIS ?WORK ?STRUCTURE))
     (instantiationOfAIS ?COPY ?STRUCTURE))

(implies
     (and
        (instantiationOfAIS ?COPY ?STRUCTURE)
          (correspondingAIS ?WORK ?STRUCTURE))
     (instantiationOfWork ?COPY ?WORK))
```

At the most general level of description, the TextLearner procedure for analyzing text involves identifying an instance of `PropositionalConceptualWork`, and reasoning about its information structure, i.e., the instance of `AbstractInformationStructure` that is its `correspondingAIS`, which, for textual works, is typically a string of characters. In short, if a document D has as its structure a string S, one fully can understand "the propositional content" of D by successfully interpreting S. Though this is a fairly innocuous view to hold at the level of whole documents, it is misleadingly dangerous when extended to smaller parts of documents, such as individual sentences, phrases or words, the meanings of which change, depending on the surrounding context supplied by

the document. Thus to talk of "the meaning" or "the right interpretation" of a *string* is misleading: A given string (e.g., "he") does not carry with it an inherent "right" meaning; rather it is a *contextualized* structure—an *occurrence* of a string within a larger structure—that can be said to have a meaning.[12]

The concept of a contextualized information structure was, in the earlier weeks of the Reading Learning Comprehension project, not explicitly represented, and only (poorly) hinted at in the Cyc ontology, indirectly through analogous type/token distinction for instances of `InformationBearingThing`. Thus the collection-term `ContextualizedInformationStructure` was reified and added to the Cyc KB. It is possible for there to be many sub-classes of contextualized information structure; however, TextLearner is only concerned with the contextualized words, phrases, and sentences, for which the CycL term `LinguisticExpressionPeg` was used. Throughout this report, "expression peg" is used to describe contextualized information structures of this type.

TextLearner uses a canonical method of representing the propositional content of a document or other target text in Cyc: It introduces, or reifies, a CycL term for the document in the "abstract work" sense of the document, and a microtheory where CycL representations of the claims forwarded in that work can be asserted. Given a CycL term for the document, such as `Chemistry-WikipediaArticle` for the Wikipedia article "Chemistry," the assertion,

```
(contextOfPCW Chemistry-WikipediaArticle
   (ContextOfPCWFn Chemistry-WikipediaArticle))
```

identifies `(ContextOfPCWFn Chemistry-WikipediaArticle)` as the microtheory where the "Chemistry" article's propositional content, if understood, would be represented. As TextLearner recognizes a distinction between the propositional and presupposed content of a document, it further reifies a microtheory for storing presupposed content:

```
(presuppositionsMtOfPCW
    (PresuppotsitionsMtOfPCWF Chemistry-WikipediaArticle)
     Chemistry-WikipediaArticle)
```

[12] Lest there be confusion on this point, the abstract information structure/contextualized information structure distinction and the type/token distinction, though analogous, are not identical. On this very page, "x" and "x" are two distinct tokens of the same type, and their relative spatial location, combined with the fact that they instantiate the same abstract information structure (the same string) is sufficient to conclude that there are two distinct contextualized information structures (two distinct occurrences of the same string) within the structure of this page. However, that does not mean that the tokens are the same thing as the contextualized structures. if one makes a photocopy of this page, one will have succeeded in generating two more distinct tokens of the same type. But just as all *four tokens* instantiate a *single* abstract information structure (the same string), each pair of tokens merely serves as a physical reminder that there are two contextualized information structures (two occurrences of that single string) within the single shared structure that both copies of this page instantiate. That there is a second pair of tokens, however, does not suggest that there is a second pair of contexualized information structures.

However, to arrive at any sort of representation of a document's content, TextLearner must analyze the *structure* of the document. To this end, it reifies an information structure functionally, using the functor `AISForFn` such that for the "Chemistry" article,

```
(correspondingAIS Chemistry-WikipediaArticle
  (AISForFn Chemistry-WikipediaArticle))
```

identifies `(AISForFn Chemistry-WikipediaArticle)` as the term denoting the actual abstract information structure of the Chemistry article—i.e., the very structure shared by all physical printings that all its physical copies must instantiate. At the same time, it reifies a "workspace" for analyzing this structure, this time applying another microtheory-denoting function:

```
(interpretationWorkspaceMtForDocument
   (InterpretingDocumentMtFn Chemistry-WikipediaArticle)
   Chemistry-WikipediaArticle)
```

Together, these structures form a microtheory hierarchy via transitive content-accesibility relation, `genlMt`. `(genlMt CONTEXT-1 CONTEXT-2)` means that every fact in the KB that is accessible in `CONTEXT-2` is also accessible to `CONTEXT-1`. The workspace microtheory, where information about the structure of the "Chemistry" article is stored, is at the bottom of the hierarchy, having a `genlMt` link up to `(ContextOfPCWFn Chemistry-WikipediaArticle)`, the context for representing the (finalized) interpretation of the article. That, in turn has a `genlMt` link up to the presuppositions Microtheory, which then connects to Cyc's commonsense layer with select genlMt links to various world knowledge microtheories. The workspace, however, hooks in to the Cyc KB at a second point, not accessible to the others: It stands in a `genlMt` link to Cyc's lexicon for English—a set of microtheories that contain information about English words, such their parts of speech, morphology, and semantics, including "denotation" mappings to CycL constants and rules for translating phrasal constructions into CycL formulae.

As part of this information structure, we have extended the CycL vocabulary for representing documents, microtheories (contexts), document-structure, and textual mentions (e.g., the third sentence of a document, the fourth occurrence of the word "nice,"), resulting in a representational framework that will support the processing of documents and subsequent reasoning about their content. As a small example of this infrastructure, we have implemented a method, in forward inference, for resolving relative URLs to absolute URLs; this enables use of hyperlinks on unknown words or names to automatically retrieve documents that will be helpful in learning their meanings.

Once TextLearner has the structure of a document fully represented, it begins processing the first paragraph, sentence by sentence, in the order in which the sentences appear. Almost all of the work TextLearner does is at the level of individual sentences; however, the model for each sentence is maintained after its completion, so that the analysis of

each new sentence can benefit from what was learned during the analysis of earlier sentences (and, conversely, so that the system can reassess some of its judgments about the content of earlier sentences in light of what is discovered about later sentences).

The first step in the sentence-analysis process is to reify a sentence-level "expression peg,"[13] which represents the occurrence of the sentence-string in the target paragraph of the selected document. In the current implementation of TextLearner, sentences are introduced through "create" operations that generate a CycL constant for the sentence-level peg. At this point, some of its defining properties are declared in the knowledge base as well. An example sentence-level peg, at this early stage, might thus have this sort of relatively bare representation, identifying it as a structure of the relevant type, and giving its relative location within `Paragraph02`, and its original html source string:

Constant: `L2R-SentenceLevelPeg-1`

```
isa: LinguisticExpressionPeg-CompleteUtterance TextStringOccurrence.

nthOccurrenceOfStructureTypeInStructure: (1 NLSentence Paragraph02).

asHtmlSourceString: "Ghandi followed a <a href="/wiki/Vegetarian"
title=Vegetarian"vegetarian</a> diet.".
```

A context model—the microtheory where almost all of TextLearner's reasoning about `L2R-SentenceLevelPeg-1` takes place—is then generated functionally, by applying the unary CycL functor `ContentModelForPegFn` to this new constant. Through forward inference, the resultant microtheory is placed "beneath" the workspace microtheory for the document via `genlMt`, allowing it access to all the content currently accessible from the workspace microtheory.

4.2.2 Tokenizations, Linkages, and Parse Trees

With the reification of a sentence-level peg and the placement of its context model in the genlMt hierarchy, TextLearner is ready to subject the sentence to syntactic analysis. To this end, TextLearner calls the free Link Parser (http://www.link.cs.cmu.edu/link/), a highly lexicalized, statistically ranked parser for English that produces 1) a tokenization of the input sentence, 2) a linkage, a graphical structure the connects tokens with syntactically and, to some degree, semantically significant links between individual tokens in the tokenization; and 3) a parse tree, the structure of which follows directly from the linkage.

[13] The use of the word "peg" here reflects a similar (but not identical) usage in the works of Bonnie Webber and others, who use "pegs" as objects from which to hang ever-increasing amounts of data as a discourse model is developed. See Webber, *A Formal Approach to Discourse Anaphora*, Doctoral Dissertation, Division of Applied Mathematics, Harvard University, 1978.

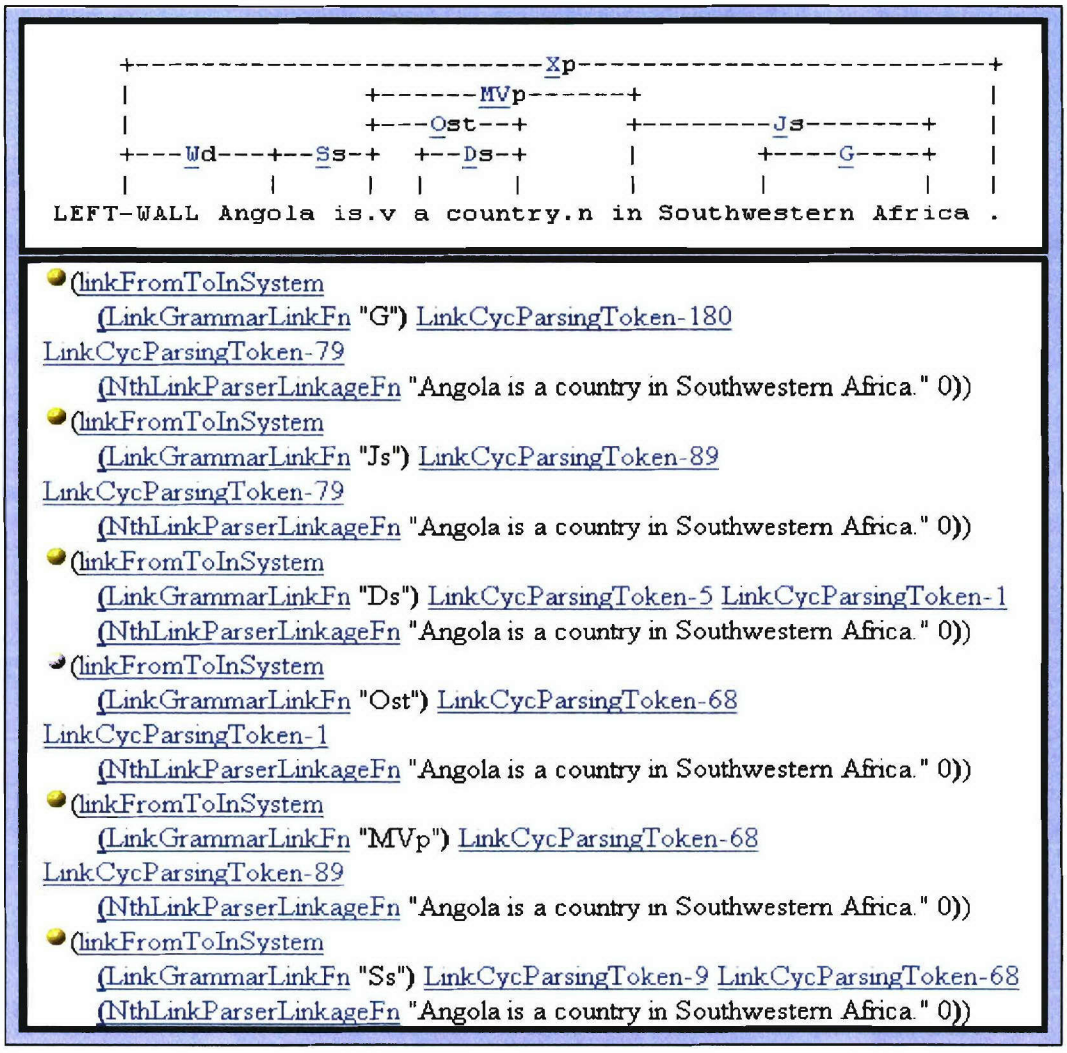

Figure 2: *Top*: a Linkage produced by the Link parser for "Angola is a country in Southwestern Africa." *Bottom*: A CycL representation of that Linkage.

TextLearner reifies all three outputs, introducing new CycL terms for the tokenization, individual tokens, the parse tree, and its nodes. Each token is associated with its string, its corresponding node in the tree, and to other tokens as they are in the linkage, via CycL predicates corresponding to Link parser links.

One reason for choosing the Link parser for use by TextLearner is that some links encode a semantic relationship, making them strictly more informative than a standard parse tree, and informative in an especially helpful way, as the semantic meanings of links can be encoded as rules for interpreting text. As will be shown in section **4.4.3**, some of this semantic information proved useful in acquiring certain rules for semantically translating new verb phrases into CycL.

During much of its development, TextLearner also called the Charniak Parser (http://www.cs.brown.edu/people/ec/#software), a broad coverage statistical parser for English, freely available for research purposes. Due to the high performance costs associated with forward inference triggered by the representation of parse trees, it was eventually decided that TextLearner should use only one syntactic parser. Link was chosen over Charniak because of the additional information it produces in the way of linkages.

The explicit representation of tokenizations, linkages, and parse trees allows TextLearner to identify semantically interesting *classes* of contextualized string-occurrences (e.g., noun phrases and verbs might be of interest) that can be detected by an analysis of the tree. Some of string-occurrences of interest are detectible via a "surface" analysis of the tree, by virtue of the tree containing an appropriately classified node (e.g., a noun phrase node). In such cases, TextLearner can sometimes augment the parse tree with information that is only implicit in the relationship of a given node to other parts of the tree. For example, the following rule is used to strengthen the classification of a node from a noun phrase to an appositive:

```
(implies
  (and
    (syntacticNodeCategory ?NODE NounPhrase)
    (syntacticNodeString ?NODE ?NODE-STRING)
    (syntacticNodeCategory ?HEAD NounPhrase)
    (syntacticNodeNthDaughter ?NODE 2 ?HEAD)
    (syntacticNodeString ?HEAD ?HEAD-STRING)
    (syntacticNodeCategory ?MOD DefiniteNounPhrase)
    (syntacticNodeNthDaughter ?NODE 1 ?MOD)
    (syntacticNodeString ?MOD ?MOD-STRING)
    (concatenateStrings ?NODE-STRING ?MOD-STRING " " ?HEAD-STRING))
  (syntacticNodeCategory ?NODE Appositive))))
```

This rule would correctly identify "the cartoon character Dick Tracy" as an appositive on the grounds that it is a noun phrase that consists of two noun phrases, the first of which, "the cartoon character" is a definite phrase that modifies the second. Knowing that a node in a parse tree is an appositive is useful for purposes of content-extraction: in general, one may conclude that the semantics of the head phrase (in this case, Dick Tracy), is an instance of the type given by the modifier (cartoon character).

A declaratively represented parse tree also supports TextLearner in the identification of string-occurrences of interest that are not reflected in the parse tree's surface structure (e.g., a tree might fail to contain a node for the appositive "the cartoon character Dick Tracy," despite having nodes for both "the cartoon character" and "Dick Tracy"; TextLearner might then conjecture the existence of the appositive based the properties of the two sibling nodes). Rather than correcting, or augmenting the tree in such cases, TextLearner simply reifies the string-occurrence of interest, thereby "extracting" it, along with any other string-occurrences of interest, from the tree, and placing it in the larger

context model. This results in a secondary tree-like structure whose "nodes" (the various reified occurrences of strings) serve as the focal points of interest as TextLearner attempts to extract content from the input text.

4.3 Methods of Content Extraction

4.3.1 The Framework: Generating SIHMs

TextLearner uses a number of methods for assigning semantic interpretations to string-occurrences, or expression pegs. Before describing each in detail, a description of the general framework by which such interpretations are added is warranted.

As noted above in **4.2**, TextLearner uses the parse trees derived from linkages to find string-occurrences of interest. These are reified as first-order objects in the context model—expression pegs—and are related to the input sentence (itself represented as an expression peg) uniquely by their string and character offset, and, where applicable, to the corresponding node in the parse tree. When a candidate interpretation `INTERPRETATION` for a reified expression peg `PEG` becomes available, TextLearner forms an hypothesis using the predicate `pegInterpretation-CycL`, where

```
(pegInterpretation-CycL PEG INTERPRETATION)
```

means that `INTERPRETATION` is the correct semantic interpretation of `PEG`.

Every assertion in the Cyc Knowledge Base is grounded, or located in a microtheory, which represents an internally consistent set of assertions—a context, so to speak—in which it is true. Most assertions about expression pegs, such as assertions describing the relationships they bear to particular strings, to nodes in a parse tree, and to each other, are located in the context model for the relevant sentence. However, `pegInterpretation-CycL` cannot be located there: The conjunction of any two `pegInterpretation-CycL` assertions for the same expression peg constitute a contradiction—they are rival hypotheses as to the correct meaning of a peg—so their co-location in a single context model would violate the internal consistency of that model.

The solution to this problem is to generate a unique microtheory for each interpretation. This has been implemented using a function, `MtWithFocalContentFn`, that takes a sentence and returns a microtheory whose "focal content" just is an assertion of that sentence. For example, the non-atomic term

```
(MtWithFocalContentFn (pegInterpretation-CycL PEG INTERP))
```

represents the microtheory whose focal content is the hypothesis the correct interpretation of the expression peg `PEG` is `INTERP`. To prevent contradiction, each microtheory generated this way is placed, via forward rules concluding to the `genlMt` relation, in a

"sibling" position, such that the contents of the relevant context model are visible to all, but the content of each is invisible to all others.

These microtheories, called Semantic Interpretation Hypothesis Microtheories, or SIHMs, provide the added advantage of serving as first-order objects in the context model. Unlike individual hypotheses (sentences), which are not (typically) reified and thus are not "fixed" in a way that would easily allow one to add assertions *about* them, microtheories are reified individuals in the Cyc KB, and so serve as a convenient "hook" on which to hang the results of TextLearner's analysis of competing interpretations. Thus the rules used to "score" possible interpretations are actually written in a way that scores them indirectly, cueing off of properties of the SIHMs that "embody" them.

4.3.2 Method I: Lexical Lookup

In addition to being a repository of common sense knowledge, the Cyc KB contains a number of hand-crafted, language-specific lexicons consisting of word-units and information about part of speech and morphology, as well as templates for translating natural language into CycL expressions.

The simplest sort of translation template in the Cyc lexicon is a simple mapping of words and parts-of-speech onto CycL terms. The predicate `denotation` is the most fundamental of these. The sentence

```
(denotation Table-TheWord CountNoun 0 Table-PieceOfFurniture)
```
[14]

means that the word "table," when used as a count noun, can map to the CycL term `Table-PieceOfFurniture`, Cyc's term for the collection of all tables. As the parse trees used available to TextLearner contain both part of speech and string information, TextLearner is able to query the lexicon for `denotation` and other, similar translation templates, to find possible semantics for lexical (leaf) nodes. If, for example, a lexical node `NODE` for the string "table" was marked as singular (a number-feature only possible for count nouns), TextLearner would look for count noun `denotation` entries for whatever word had "table" as its singular form—i.e., `Table-TheWord`—and would find `Table-PieceOfFurniture` listed as an appropriate possible interpretation. *Lexical ambiguity* arises when a single word and part of speech pairing can map to multiple CycL terms. For example, the existence of the additional assertion,

```
(denotation Table-TheWord CountNoun 1 Table-VisualCW)
```

means that the word "table," when used as a count noun, can also map to the CycL term `Table-VisualCW`, Cyc's term for the collection of graphical tables, as one might see in an earnings report or scientific study. Given the presence of both assertions in the Cyc

[14] The "0" is an artifact of an early attempt in the development in Cyc to uniquify word senses, and has no real significance in this context.

lexicon, TextLearner would construct two competing hypotheses as to the correct interpretation of the occurrence of "table" in question.

The Cyc Lexicon is not comprehensive for English, so the opportunity arises for TextLearner to 1) learn new words when it encounters a lexical node that does not correspond to an English word with which it is familiar, and 2) learn new `denotation` assertions for existing words where such assertions are lacking. In response to the first sort of gap discovery, TextLearner reifies a new word with the string and part of speech features given by the parse tree. This effectively transforms situations of type 1) into situations of type 2)—that is, having introduced a word to fill the type 1) gap, TextLearner finds itself in a situation where it has the appropriate word, but some `denotation` assertions are missing. TextLearner contains several techniques for positing new possible denotations under these circumstances. In this section we discuss two: The use of Noun Learner technology and a method of "reading" WordNet entries. A third technique, the use of semantic closeness of known denotations for a given word and part of speech to suggest new denotations for that same word and a different part of speech, is more accurately characterized as a rule-acquisition technique, and so is discussed in section **4.4**, below.

Noun Learner

The Noun Learner is a technology developed under a previous Cyc project that uses a verb/argument analysis of a corpus to learn new nouns. Given a new noun, say "derivation," the Noun Learner would check the corpus to see what verbs it occurred as a subject and what verbs it occurred as an object. For each "subject verb" found, the Noun Learner would compile a list of nouns that occur in that corpus as a subject, doing the analogous procedure for each "object verb." Noun Learner then attempts to map each noun into Cyc, via an appeal to denotation assertions in the Cyc lexicon. The result is a set of CycL concepts for which the Noun Learner attempts to find a close generalization that covers, if not all concepts in the set, those with the greatest statistical frequency. The end result is a small number of collections (typically one) that the system can suggest as a defining `genls` for the new noun. Keeping with the "derivation" example, the Noun Learner's analysis would suggest `CreationEvent`, the collection of all events in which something is created, as a defining genls for the collection of derivations. That is, according to the Noun Learner analysis, a derivation is a type of creation event.

In the context of TextLearner, the appeal to Noun Learner unfolds in the following way: Having encountered a lexical node for an occurrence of the count noun "derivation," TextLearner would appeal to the Cyc lexicon and find `Derive-TheWord`, for which "derivation" was known to be the singular form, but for which there is no denotation. TextLearner would then reify a new term `Derivation`, about which we know virtually nothing, and add a gap-filling denotation to the Cyc KB:

```
(denotation Derive-TheWord CountNoun 0 Derivation)
```

At the same time, it would query the Noun Learner via forward inference, giving it the string "derivation" as an input. Noun Learner would respond with `CreationEvent`, as described above, which TextLearner would use to generate the assertion

```
(genls Derivation CreationEvent)
```

which it places in a special Noun Learner "conjectures" microtheory for later review by a trained ontologist.[15] In the meantime, TextLearner has added a semantic target for a lexical node and can proceed analyzing the rest of the text.

WordNet

A second method for filling in lexical gaps involves having TextLearner "look up" a new word in a dictionary (specifically, the online and freely available WordNet[16]) and use the information it finds there to define the new concept. Specifically, when encountering a new noun a forward rule generates a query to the WordNet 2.1 database, the connection brokered by a PostgreSQL server. This query returns a set of CycL terms, each of which is arrived at by mapping into CycL each item in the synset and immediate hypernym lists for that word.[17] As there will frequently be a one-to-many mapping from words in the synset and hypernym lists to CycL terms, two measures are taken to select the most appropriate CycL. The first is a syntactic measure: The part of speech information on the new noun, which has been recorded as a property of the relevant node in the parse tree, is used as a filter, so that if the new noun occurs as a count noun and a word (e.g., "fish") appears in the synset or hypernym list, only count noun interpretations will be considered (so that `Fish`, Cyc's term for the collection of fish, will be returned and `(MeatFn EdibleFish)` will not). The second is a semantic measure: All possible sets of unambiguous interpretations of the synset and hypernym words are first generated and compared as to how mutually *similar* the elements of those sets are. The set whose members are the most similar is selected. Similarity is based on the ontology, and is computed as the intersection of the number of generalizations of each element in the set over the union of their generalizations, where a generalization of an element includes every collection such that element is either an instance or specialization of that collection. With these filters in place, TextLearner is able to extract a possible `genls` definition for each new word that it encounters that is also found in WordNet. For example, if the noun is "vacillator," TextLearner will add a denotation to the lexicon,

```
(denotation Vacillator-TheWord CountNoun 0 Vacillator)
```

---

[15] In the course of the Rapid Knowledge Formation program, Cyc has developed the ability to form questions that can help guide an ontologist in fleshing out what "stub" concepts such as Derivation might mean. For example, since axiomatically, every creation event stands in a outputsCreated relation to something that is created, Cyc would form the question, "what kinds of things are created in a derivation?" or (relationAllExists outputsCreated Derivation ?WHAT).

[16] See http://www.cogsci.princeton.edu/~wn/.

[17] There will be many WordNet entries for a single word, each purportedly representing a distinct sense. For the gap-filling strategy described here, TextLearner considers the first entry, which reflects the most frequently used sense.

and then query WordNet for the syntactically and semantically best set of CycL reflected in the synset and hypernym lists for "vacillator." In this case, as the WordNet synset for which is "waverer, vacillator, hesitator, hesitater," the query will return a set of CycL terms that includes `(FrequentPerformerFn Hesitating)`, the CycL term for "hesitator." As with the Noun Learner strategy, TextLearner responds by generating an assertion

```
(genls Vacillator (FrequentPerformerFn Hesitating))
```

that it adds to a special "WordNet conjectures" microtheory for later review by a trained ontologist. Because there is no guarantee that the WordNet and Noun Learner strategies will yield mutually consistent results, the respective microtheories containing their respective genls assertions do not stand in any sort of `genlMt` relationship, so that the contents of the one are inaccessible from the other.

4.3.3 Method II: Phrase Structure Parser

First developed under the Rapid Knowledge Formation project, the Cyc Phrase Structure Parser (PSP) is a chart parser that takes a bottom-up approach to constructing semantics for complex phrases. TextLearner calls the PSP as needed to generate SIHMs for phrase-level pegs encountered while processing the input text. As with the assignment of semantics to lexical nodes, this is done through forward rules. To make this possible, vocabulary was developed for this project that would allow it to query Cyc for parser ouput. Specifically, the arity-4 predicate

```
parserRunToSpecificationWithInputStringReturnsCyclification
```

allows one to identify a parser (such as the PSP), a specification as to how that parser should be run, and an input string, and query the Cyc inference engine for the output semantics. Additional vocabulary was developed to allow for easy, on-the-fly reification of possible parser specifications. For the PSP, the specifications of interest were variants of its default settings with respect to the *part of speech* (e.g., singular, infinitive, regular degree, etc.) and *phrase category* (verb phrase, noun phrase, adjectival phrase, etc.) parameters, the appropriate values for which are transparent from TextLearner's context model.

```
(SpecificationVariantFn
   (DefaultParameterSpecificationFn CycPhraseStructureParser)
   parserSpecificationPOSPredicate singular-Generic)
```

denotes the specification of the PSP that differs from the default in that the input string will be parsed using `singular-Generic` as a setting.

Given this sort of representation, TextLearner is able to query the PSP using any of these specifications as appropriate.

4.3.4 Method III: Anaphora Resolution

Anaphora resolution is the process of determining the antecedent of an expression, such as a pronoun or definite noun phrase, whose meaning is dependent on the meaning of that antecedent. Because TextLearner draws upon the Cyc Knowledge Base, it is in a position to bring both syntactic and semantic considerations to bear in resolving anaphora.[18] For example, using syntactic methods, one can rule out "my brother" as the antecedent of "they" on the grounds that that the former is singular while the latter is plural. However, there are no syntactic grounds for ruling out "my brother" as the antecedent of "my sister." However, there are semantic grounds available: "my brother" must refer to the speaker's/author's male sibling, and as no female is a male, it must be that "my brother" is not the antecedent of "my sister."

In TextLearner, anaphora resolution proceeds by identifying candidate antecedents of anaphoric expressions—pronouns and definite noun phrase—which results in the generation of `pegInterpretation-CycL` SIHMs for those expressions. As information comes in some candidate antecedents are removed, in a process of truth-maintenance described above in section **3.1.2**. The assignment of semantics to anaphoric pegs is always linked to specific SIHMs for the antecedents. For example, consider a situation where ANA is a reified expression for "the President," and ANTE is an earlier reified expression for "Bush." ANTE might have two incompatible hypotheses

```
(pegInterpretation-CycL ANTE GeorgeWBush)

(pegInterpretationCycL ANTE Bush-Shrub)
```

only one of which—the first—licenses treating ANTE as an antecedent for ANA. Specifically, `GeorgeWBush` is an instance of `PresidentOfOrganzation`, the candidate semantics of the reified peg for "President," the head of the definite expression ANA, whereas `Bush-Shrub` and `PresidentOfOrganization` are disjoint. So the decision to conclude that ANTE is a candidate antecedent for ANA is done relative to a SIHM for ANTE, allowing TextLearner to generate a *single* semantic hypothesis

```
(pegInterpretation-CycL ANA GeorgeWBush).
```

This, in fact, reinforces the `GeorgeWBush` interpretation of ANTE: One of the strategies for scoring SIHMs involves rewarding an interpretation if the CycL semantics it promotes appear in multiple SIHMs corresponding to different strings. `Bush-Shrub` appears as candidate semantics for n many pegs with a single string ("Bush") whereas `GeorgeWBush` appears for those same pegs plus pegs determined to be other ways of

[18] The methods of anaphora resolution implemented in TextLearner are an extension of the work started described in Jon Curtis, G. Matthews, D. Baxter, "On the Effective Use of Cyc in a Question Answering System" in *Proceedings of the IJCAI Workshop on Knowledge and Reasoning for Answering Questions*, pp. 61-70, Edinburgh, Scotland, July 30, 2005.

referring to him, such "George Bush," "George W. Bush," and, notably in this instance, "the President."

4.3.5 Method IV: Cyclifier

The cyclifier is a program designed to map Natural Language sentences into sentences of CycL. Within TextLearner, the Cyclifier uses two methods. First, it calls a syntactic parser (the link parser) on the input sentence in order to obtain grammatical components (e.g, the main verb, its subject and direct or indirect objects), and then accesses the Cyc Lexicon for both word level templates (e.g., `denotation` assertions) and phrase level templates known as semantic translation templates. Semantic translation templates identify the CycL relations that relate the CycL denotations of the various grammatical components. For example, the adjective semantic translation template,

```
(adjSemTrans Employ-TheWord 0 RegularAdjFrame
             (employeeStatus :NOUN Employee))
```

means that one can translate certain adjectival uses of "employed" into a sentence that includes a clause relating the semantics of the modified noun to `Employee` via the relation `employeeStatus`. So "Bob is employed" would translate straightforwardly into `(employeeStatus Bob Employee)` and a sentence that included the phrase "employed person" would translate to a quantified CycL sentence that attributes to some variable, constrained to be an instance of Person, that the, e.g.,

```
(… (and … (isa ?PERSON Person) (employeeStatus ?PERSON Employee) …)))
```

In addition to adjectives, The Cyclifier makes use of semantic translation templates for nouns, verbs, adverbs, and prepositions. Verb semantic translation templates, which the TextLearner system acquires as it encounters new verb constructions, will be described in a bit more detail in Section **4.4.3** below, as part of the larger discussion of rule acquisition.

The second method that the Cyclifier uses to generate sentence-level semantics is to use sentence-level templates, often generated by hand, but more recently from example, that map arbitrarily complex English to CycL sentences without making the same assumptions about compositionality that lie behind the semantic translation template method. The sentence-level template method allows one to effect compact, even partial semantics of a class of sentences.

For example, one might wish to be able to extract from, "Those familiar with the Cyc project will doubtless recognize the name Doug Lenat, its founder" to `(founderOfProject Lenat CycProject)`. The existence of a template that effectively "eats up" the peripheral information about the audience ("those familiar …") or about the speaker's certainty ("doubtless"), would enable this extraction, where a series of phrase-level semantic translation templates, unless it there were a template available for every

phrase, could not. As will be seen below in the discussion of rule-acquisition, TextLearner has some capabilities for generating such "partial" templates as it processes text.

The Cyclifier is the last content extraction component called inside TextLearner, which allows it to benefit from the addition of any new lexical entries (such as semantic translation templates, denotations, or sentence-level templates) it acquires. Of special interest is the way in which the Cyclifier is able to make use of anaphora resolution. Because it operates by accessing the lexicon for word-to-CycL mappings and uses its own internal parse tree and tokenization to identify words and grammatical units, the resolution of anaphora done at an earlier stage is not thereby automatically available. That is, TextLearner's resolution of some anaphoric expression (say, an expression peg for "he") to a given antecedent (say, an expression peg for "John Kerry"), results in an assertion linking those two and a SIHM hypothesizing that the semantics for this occurrence of "John Kerry" are those of this occurrence of "he." This information isn't reflected anywhere in the Cyclifier's internally-generated parse tree.

As the Cyclifier assigns semantics to the constituents of its internal tree by appealing to the lexicon, the TextLearner makes the results of anaphora resolution available by making temporary entries to the lexicon. For example, if an occurrence of "he" is deemed to resolve to an earlier occurrence of "John Kerry" for which `JohnKerry` is the semantics, TextLearner will add

```
(denotation He-TheWord PersonalPronoun 0 JohnKerry)
```

to the lexicon. When the Cyclifier then has an opportunity to parse the original sentence, it has a semantic target for "he"—`JohnKerry`—readily available, that reflects the results of the anaphora resolution. The entry is tagged as temporary (by being placed in a "temporary" lexical microtheory) and is removed when TextLearner is done processing the sentence, thereby preventing uses of that pronoun in later sentences from being interpreted as referring to John Kerry unless anaphora resolution mechanisms license it. Though far from a perfect solution—multiple, correct resolutions of the same pronoun will result in unwanted ambiguity this way—it is a workable first approximation.

When the Cyclifier is finished, each interpretation it proposes is recorded as a SIHM, along with all the phrase- and lexical-level interpretations.

4.4 Methods of Rule Acquisition

4.4.1 Lexical Gap-Filling Rules

In cases where the Cyc lexicon contains denotations for some, but not all, parts of speech for a word, TextLearner can draw upon the semantic space surrounding a denotation to find candidates to fill the gap. For example, `Beer-TheWord` is known to have both a

count noun form ("beer" and "beers") and a mass noun form ("beer"), but the Cyc lexicon only contains a single denotation handling only the mass noun case.

```
(denotation Beer-TheWord MassNoun 0 Beer)
```

This leaves a gap in the lexicon with respect to any count noun denotation for `Beer-TheWord`. In the context of TextLearner, this means that when considering an occurrence of "beer" that has been identified with a lexical parse tree node marked as having mass number, the lexicon will be unable to provide a possible semantic interpretation. an attempt to fill this gap, TextLearner examines the relationship of Beer to other concepts in the Cyc ontology, not only to fill this particular gap, but also to hypothesize rules that can be used to fill similar gaps of that type (i.e., count noun gaps when a mass noun denotation is known).

For example, Cyc knows the following salient fact about beer:

```
(servingTypeOfFoodOrDrinkType (FoodServingFn Beer) Beer).
```

This identifies `(FoodServingFn Beer)` as the collection of individual beer servings. TextLearner then forms a general hypothesis to explain how the count noun `denotation` gap of `Beer-TheWord` might be filled: The relation `servingTypeOfFoodOrDrinkType` is a lexical-extension relation, such that when a concept denoted by a mass noun appears in the second argument, the other argument will contain a concept denoted by a count noun version of that noun. This rule will forward-derive a new denotation for `Beer-TheWord`:

```
(denotation Beer-TheWord CountNoun 0 (FoodServingFn Beer))
```

as well as denotations for any other mass noun denotations for which a `servingTypeOfFoodOrDrinkType` link is known, such as "soda," "pizza," "water," "spaghetti," and the like. These rules are placed in a special microtheory for later review by trained ontologists, but the `denotation` entries they generate are made immediately available for TextLearner to use.

4.4.2 Noun Compound Rules

TextLearner's first step in interpreting a noun compound involves recognizing cases where a parse-tree includes two adjacent nouns, the first of these having a non-plural form (e.g. "noun compounds" or "Bantu word"). On the basis of such configuration, the system concludes that the relation `candidateNounCompound` holds between the first node,—we'll call this the noun compound modifier—and the second node, the noun compound head. This `candidateNounCompound` assertion, in turn, serves as a (partial) basis for concluding a variety of other assertion· that are used to drive both noun-compound interpretations and the hypothesizing of new noun compound rules.

Noun Compound interpretation

When the system recognizes that the `candidateNounCompound` relation holds between nodes in a tree, it determines whether the syntactic and semantic features of these nodes are compatible with the syntactic and semantic constraints of noun compound rules familiar to the system. On the basis of this determination, some number of `nounCompoundWithRule` assertions will be generated via forward inference. `nounCompoundWithRule`, a quintary (arity 5) predicate, relates the modifier and head nodes, along with their relevant CycL interpretations, to whatever rules they happen to satisfy.

For a noun compound like "faculty investigation," this process includes determining whether any potential interpretations of "faculty" and "investigation" are consistent with any known rules. In this case, we find that when the modifier is interpreted as `AcademicDepartment` and the head as `Investigation`, these nodes satisfy the constraints of `CourtRuling-NCR`, a rule that can be used to relate specializations of `IntelligentAgent` to specializations of `PurposefulAction` generally.

The actual assignment of a CycL interpretation is handled by the Phrase Structure Parser, described in section **4.3.5**, above.

Noun Compound Rule Hypotheses

The above-mentioned `candidateNounCompound` assertions also trigger attempts to derive new (hypothesized) noun compound rules on the basis other (generally non-linguistic) knowledge in the KB. Currently, TextLearner's strategies for doing this capitalize on knowledge that has been encoded using generalizations such as "every word is a word in some language," or, in CycL:

```
(relationAllExists wordInLanguage LexicalWord NaturalLanguage)
```

Using such assertions, TextLearner is able to hypothesize rules like:

```
(NCRuleWithTemplateFn
  (TheList SubcollectionOfWithRelationToFn)
    (TheList TheNCHead genls LexicalWord) wordInLanguage
    (TheList TheNCModifier isa NaturalLanguage)))
```

The first of these predicts that a noun that denotes a Natural Language (e.g "Bantu") may be followed by a noun which denotes a type of word (e.g. "word" or "name"). It further predicts that a compound which has this form will denote terms of the type specified by the head which are also words of the language specified by the modifier. That is, when TextLearner encounters a noun compound "Bantu word," it will use this rule to interpret it as possibly meaning the collection of words in the Bantu language, or

```
(SubcollectionOfWithRelationToFn LexicalWord wordInLanguage
                                 BantuLanguageSet)
```

4.4.3 Semantic Translation Templates

Semantic translation (sem-trans) templates are CycL sentences that give a recipe for translating natural language phrases into CycL. These templates are used by cyclification (NL-to-CycL) parsers, such as the PSP and the Cyclifier. The performance of these parsers are thus to a large degree limited by issues of coverage: A particular semantic translation template needed to handle a given use of a verb might be missing, leading the parsers to produce partial or no results. Given that TextLearner calls upon the PSP and Cyclifier to generate candidate semantic interpretations, its ability to extract information from text will also be limited by coverage, making the acquisition of new templates part of its "learning reading" mandate.

Some progress has been made in the area of acquiring new verb semantic translation templates in response to encountering prepositional complements. Consider, for example, the sentence

"The heart pumps blood to the lungs."

The main verb of this sentence, "pumps," has an entry in the Cyc Lexicon under `Pump-TheWord` for which the following semantic translation templates are known:

```
Mt: GeneralEnglishMt
(verbSemTrans Pump-TheWord 1 TransitiveNPFrame
      (and
          (isa :ACTION PumpingFluid)
          (providerOfMotiveForce :ACTION :SUBJECT)
          (primaryObjectMoving :ACTION :OBJECT)))

(verbSemTrans Pump-TheWord 1 IntransitiveVerbFrame
      (and
          (isa :ACTION PumpingFluid)
          (primaryObjectMoving :ACTION :SUBJECT)))

(verbSemTrans Pump-TheWord 0 TransitiveNPFrame
      (and
          (isa :ACTION Pumping-MakingSomethingAvailable)
          (performedBy :ACTION :SUBJECT)
          (objectActedOn :ACTION :OBJECT)))
```

This use of `Pump-TheWord` is transitive ("The heart pumps blood ...") and so there is enough information in the `TransitiveNPFrame` translation template to cover the subject-verb-object relationship; however, there is insufficient information to cover the prepositional complement. In other words, the most complete translation Cyc can generate using its lexical knowledge of `Pump-TheWord` is one that drops the PP complement altogether:

```
(thereExists ?HEART
 (thereExists ?PUMPING
  (thereExists ?BLOOD
   (and
    (isa ?HEART Heart)
    (isa ?BLOOD Blood)
    (isa ?PUMPING PumpingFluid)
    (primaryObjectMoving ?PUMPING ?BLOOD)
    (providerOfMotiveForce ?PUMPING ?HEART)))))
```

In other words, the cyclifier effectively parses the sentence as though it were "The heart pumps blood." To overcome this deficit, TextLearner does an analysis of the reified linkage for this sentence, which shows that

1 "pumps" is the main verb with "blood" as its object
2 "to the lungs" is a prepositional complement, with "to" as the relevant preposition

1 gives enough information for TextLearner to know that the semantic translation template for `Pump-TheWord` using the transitive NP comp frame is relevant. Thus is can use that template as the basis for building a new template that will allow it to handle the full verb phrase. **2** gives enough information for TextLearner to apply a common-sense based heuristic for interpreting the PP complement. In particular, the Cyc lexicon contains a mapping from English prepositions to "underspecified" CycL predicates that serve as semantic stubs for the interpretation of those prepositions. In this case, Cyc knows that `To-TheWord` maps to the predicate `to-UnderspecifiedLocation`. These underspecified predicates are extremely general, living at the top of a vast predicate hierarchy and have dozens (in some case hundreds) of specializations, each of which is a possible interpretation for any given use of a preposition. Cyc's strategy for building a new semantic translation template is to do a focused search, downward through the hierarchy of predicates under `to-UnderspecifiedLocation` to find the best possible predicate. To focus the search, Cyc applies commonsense knowledge about `PumpingFluid`, a possible denotation for `Pump-TheWord` already known to be applicable from the semantic translation template known to be relevant from **1**.

`PumpingFluid` is the Cyc term for the collection of all fluid-pumping events. To find a suitable role for "to," Cyc looks for a specialization of `to-UnderspecifiedLocation` that is the most-specific, required role for any fluid-pumping event. Since every fluid-pumping event is a location-change event, i.e.,

```
(genls PumpingFluid Translocation)
```

and every location-change event has a location that is the "destination" of the object moving in the event, the role `toLocation` is required for any fluid-pumping event. As there are no more specific predicates in the `to-UnderspecifiedLocation` hierarchy that

are also required, `toLocation` is selected. Thus Cyc combines the template for the transitive frame for "pump"

```
(and
   (isa :ACTION Pumping-MakingSomethingAvailable)
   (performedBy :ACTION :SUBJECT)
   (objectActedOn :ACTION :OBJECT)))
```

with a clause for handling the PP frame

```
   (toLocation :ACTION :OBLIQUE-OBJECT)
```

to construct a new semantic translation template

```
Mt : GeneralEnglishMt
(verbSemTrans Pump-TheWord 311
     (PPCompFrameFn DitransitivePPFrameType To-TheWord)
     (and
         (isa :ACTION PumpingFluid)
         (providerOfMotiveForce :ACTION :SUBJECT)
         (primaryObjectMoving :ACTION :OBJECT)
         (toLocation :ACTION :OBLIQUE-OBJECT)))
```

This template allows the cyclifier to produce a more complete parse, that covers all of the components of the target sentence:

```
(thereExists ?PUMPING
 (thereExists ?HEART
  (thereExists ?BLOOD
   (thereExists ?LUNGS
    (and
     (isa ?PUMPING PumpingFluid)
     (isa ?HEART Heart)
     (isa ?LUNGS Lung)
     (isa ?BLOOD Blood)
     (primaryObjectMoving ?PUMPING ?BLOOD)
     (providerOfMotiveForce ?PUMPING ?HEART)
     (toLocation ?PUMPING ?LUNGS)))))
```

Here's the full CycL for the rule used in this example:

```
(implies
  (and
     (underspecifiedPredicateForPreposition ?UND ?PREP)
     (genlPreds ?PRED ?UND)
     (requiredActorSlots ?TYPE ?PRED)
     (termFormulas ?TYPE ?X)
     ((LinkGrammarLinkFn "O") ?VL ?OL)
     ((LinkGrammarLinkFn "MV") ?VL ?PL)
     (tokenString ?VL ?VSTRING)
     (linkageTokenization ?LINKAGE ?TOKENIZATION)
     (tokenString ?PL ?STRING)
```


```
    (tokenTokenization ?OL ?TOKENIZATION)
    (wordForms ?WORD verbStrings ?VSTRING)
    (ppCompFrameForTypeAndWord ?FRAME DitransitivePPFrameType ?PREP)
    (wordForms ?PREP prepositionStrings ?STRING)
    (argN ?ARG1 1 ?X)
    (argN ?ARG2 2 ?X)
    (argN ?ARG3 3 ?X)
    (unknownSentence
      (thereExists ?BETTER
        (and
          (genlPreds ?BETTER ?PRED)
          (different ?BETTER ?PRED)
          (requiredActorSlots ?TYPE ?BETTER))))
    (denotation ?WORD Verb ?M ?TYPE)
    (verbSemTrans ?WORD ?N TransitiveNPFrame ?X)
    (concatenateStrings ?NUM-STRING "3"
         (IntegerToStringFn ?N)
         (IntegerToStringFn ?M)))
(semTransExtractedFromStructure
   (MakeFormulaFn verbSemTrans
        (TheList ?WORD
           (StringToIntegerFn ?NUM-STRING) ?FRAME
              (MakeFormulaFn and
                (TheList ?ARG1 ?ARG2 ?ARG3
                   (?PRED :ACTION :OBLIQUE-OBJECT))))) ?LINKAGE))
```

The `StringToIntegerFn` and `IntegerToStringFn` clauses are merely there to guarantee a new sense number for the `verbSemTrans` rule to be constructed. Having concluded that `semTransExtractedFromStructure` holds between the new template and the linkage structure that was analyzed in order to produce it, a second rule concludes that `semTransExtractedFromStructure` also holds between the template and the text from which this linkage was built. Where the language of the input text is known, this gives Cyc enough information to assert the template into the lexicon for that language:

```
(implies
      (and
          (languageHasRootLexicon ?LANGUAGE ?LEXICON)
          (semTransExtractedFromStructure ?CYCL ?AS)
          (structurePrimaryLanguage ?AS ?LANGUAGE))
      (ist ?LEXICON ?CYCL))
```

4.4.4 EBMT Templates

As noted in section **4.3.5** above, one method by which TextLearner proposes semantics for input sentences is through the application of sentence-level templates. Due to some recent work in the area of Example-based Machine Translation (EBMT), Cyc is now able to construct such templates by example, so that if a complete English-to-CycL mapping is presented, it can generalize the example into a template. EBMT traditionally has been used to effect translations from one natural language to another, by comparing large

parallel corpora in different languages.[19] In TextLearner, Cyc can acquire such templates by identifying high-scoring sentence-level interpretations and submitting the original English and the interpretation as a example mapping pair. Another method used is to extract a conjunction of provable clauses from an interpretation, the result of which is a high-scoring partial interpretation (high scoring because Cyc has proved it true), and submit this constructed interpretation along with the original English as an EBMT mapping example. For example, where the Cyclifier might return the following, partially correct CycL for "Angola is a country in Africa that has a long history of violence."

```
(thereExists ?STORY
  (thereExists ?ANGOLA
    (and
      (isa ?ANGOLA Country)
      (equals ?ANGOLA Angola)
      (inRegion ?ANGOLAContinentOfAfrica)
      (isa ?STORY HistoricalNarrative)
      (duration ?STORY LongTime)
      (topicOf ?STORY ViolentAction)
      (hasRightsOver ?ANGOLA ?STORY))))
```

Though the translation of the entire sentence is unsatisfactory—it claims that there is some story about violence that Angola has rights over—there are parts of it, namely that *Angola is a country* and *Angola is in Africa* that are correct, because Cyc can prove both literals given its existing knowledge of geopolitics. TextLearner in this case would pull both literals out and combine them, proposing this partial mapping that can be used as an EBMT template:

```
(thereExists ?ANGOLA
   (and
    (isa ?ANGOLA Country)
    (equals ?ANGOLA Angola)
    (inRegion ?ANGOLA ContinentOfAfrica)))
```

The EBMT code then would generalize the mapping into a template for rapidly interpreting sentences of the form "COUNTRY is a TYPE in REGION that ..." which would be sufficient from extracting at least the type and location of a geopolitical entity, even if the relative clause cannot be understood.

4.5 Knowledge Finalization

TextLearner's knowledge-acquisition "target"—content extracted from documents and rules for enabling better content extraction—make up only a small fraction of the data TextLearner generates. For a single sentence, TextLearner can generate thousands of assertions, most of which—assertions about tokenizations, linkages, trees, and expression-pegs, in particular—are of little benefit to TextLearner apart from the support they provide during specific, one-time content extraction and rule acquisition events.

[19] See, for example, http://www-2.cs.cmu.edu/afs/cs.cmu.edu/user/ralf/pub/WWW/ebmt/ebmt.html, Ralf Brown's research page.

Moreover to include all of this data into the Cyc KB permanently would unnecessarily bloat its size. Thus, after each sentence it processes, TextLearner executes a knowledge finalization script, where the "wheat is separated from the chaffe" so-to-speak, so that a file containing all and only the useful "target" content is generated and included in Cyc permanently.

After this finalization step is complete, TextLearner performs a "distillation" process, whereby the results of disambiguation can be recorded without requiring that the weighty infrastructure used to generate those results be maintained or transmitted to the full Cyc KB. Specifically, queries are run that produce assertions associating strings, offsets, CycL semantics, and TextLearner's score for every expression-peg (including sentences) for which a possible interpretation was generated. As this process is performed after each sentence, the scoring process is also re-performed, enabling the TextLearner to update its "opinion" as to the correct interpretations of previous sentences, words, and phrases, in response to having processed subsequent sentences. This distillation process also involves generating a similar record of extracted presuppositions and their scores. The net result of the distillation process is a detailed semantic annotation of the document, which can be used by other applications for question-answering or information retrieval.

The TextLearner knowledge finalization process has been represented in the KB as a script, `FinalizingKnowledgeAcquiredByTextLearner`. This allows TextLearner to query for the scenes of the script, in order, allowing it to retrieve the queries and the order to run them in. This also makes the finalization process something that can be modified in the Knowlege Base.

5 Technical Results

5.1 Disambiguation Experiment

A significant challenge to any autonomous reading system is correctly identifying the referents of ambiguous words and phrases, and assigning confidences to such identifications. Early and reliable identification of the concepts referred to in the document being read allows the system to draw stronger conclusions about what is being said about them, and also allows it to avoid an explosive amount of unnecessary extra work entertaining competing hypotheses. For instance, if a sentence contains two ambiguous expressions with three possible interpretations apiece, the number of possible interpretations of the sentence is nine times what it would be if the referent of each of those two expressions were identified with high enough confidence to ignore the other possible referents.

To improve our system's ability to reduce lexical ambiguity, we selected a set of four Wikipedia articles on diverse topics—Angola, Thomas Jefferson, backgammon, and biology—identified the words that our lexicon treated as ambiguous, and devised a number of strategies to reward certain interpretations over others. The strategies made heavy use of the Cyc ontology to reward interpretations using two semantic criteria:

semantic closeness and semantic contribution. Semantic closeness strategies reward pairs of hypothesis (SIHMs) when the CycL interpretations they posit stand in any number of specified relationships in the Cyc knowledge base. For instance, interpretations of a set of nearby words that are all instances of the same collection would be rewarded this way. Other semantic closeness strategies rewarded interpretations of expressions as instances of collections denoted by nearby expressions, or as denoting related concepts, such as a collection of things or people and a collection of events they are known to be capable of performing.

Augmenting this sort of disambiguation procedure, semantic contribution strategies reward an interpretation when it literally occurs inside a semantic interpretation of an expression that subsumes the expression-peg in question. For example, the word "book" is ambiguous between an *authored work* (as in, "I've read all her books") and a *book copy* ("The book I borrowed is on the table."). However, in the context of the phrase, "worn book," only one interpretation is possible—the physical book copy, which, unlike a conceptual work, can undergo wear and tear. In text learner, this would mean rewarding the book copy interpretation—`BookCopy`—on the grounds that there is an interpretation for "worn book"—`(SubcollectionOfWithRelationToFn BookCopy physicalStructuralFeatures Worn)`—in which `BookCopy` appears.

We parameterized all these strategies, and ran a hill-climbing algorithm to optimize the parameter values for best performance on our test documents. The strategies, parameters, and optimized values are all stored in the knowledge base, and can be modified or added to on the basis of future efforts, ideally by the autonomous operation of the system itself. The results of this experiment are published in Curtis, *et al*, "**On the Application of the Cyc Ontology to Word Sense Disambiguation**," *Proceedings, FLAIRS 2006*, 652-657.

The below chart summarizes the results of the disambiguation experiment. Among the four articles processed, the average number of candidate possible CycL interpretations for an ambiguous word was 3. Just over 71% of the time the correct concept was included in the set of candidate denotations. Left to chance, disambiguation could be performed correctly across the whole corpus 33.49% of the time; however, the TextLearner method for disambiguation performed correctly 56.61% of the time. This reflects a 69.04% improvement over chance.

Document	Mean Size	Percent Usable	Chance Success Rate	System Success Rate	Percent Over Chance
"Angola"	2.96	82.67%	33.83%	53.64%	58.56%
"Backgammon"	3.29	61.80%	30.43%	53.80%	76.80%
"Biology"	2.62	79.21%	38.20%	68.85%	80.23%
"Jefferson"	2.99	73.66%	33.40%	53.23%	59.37%
Cumulative	3.00	71.52%	33.49%	56.61%	69.04%

5.2 Rate of Acquisition Experiment

Near the end of the Reading Learning Comprehension project, a second experiment was set up to gather data about TextLearner's rate of acquisition of new concepts and rules. To this end, Text Learner was "fed" the complete text of the Wikipedia article "Chemistry,"[20] which contains 97 sentences, with the goal of determining a per sentence rate of new concept and rule extractions.

Several technical difficulties (including a company-wide hardware changeover) resulted in multiple re-starts. This delay was compounded by general forward-inference slowness, which was itself compounded by a "snowball" effect, in which forward inference slows as the number of context models—one for each sentence, each containing thousands of new assertions—increases. At the time of writing this report, TextLearner had finished processing three paragraphs, a total of 11 sentences, from which 7 new concepts and 1 new word (but, surprisingly, no rules) had been extracted. This rate of .64 extractions per sentence appears promising, but cannot be viewed as a true reflection of TextLearner's acquisition capabilities, given the small sample size. The processing of the "Chemistry" article continues, however, as will a general (i.e., not-specific to this project) investigation into how Cyc-based applications can be insulated against this forward inference slowness "snowballing."

6 Findings and Conclusions

At present, TextLearner's ability to generate higher quality interpretations than previous NL-to-CycL transformation technologies is mostly evident in how it performs against the numerous use- and test-cases that were used to drive its development. The slowness of the system, as noted in the discussion of the "Chemistry" experiment, has been the biggest obstacle to gathering large amounts of data as to TextLearner's capabilities. Nevertheless, TextLearner has been deployed at Cycorp as a sort of "junior ontologist," where we expect that, with occasional monitoring, it will succeed in steadily (albeit slowly) working through online articles, extracting new facts and rules, and submitting the distilled results of its processing for inclusion in a full Cyc build. If later opportunities to work on this technology present themselves, a first order of priority will be to find ways to speed up aspects of the system, using both time-tested techniques (such as replacing theorem-proving steps with efficient "heuristic-level" reasoning modules) and techniques new to Cycorp, such as parallelism. Some aspects of TextLearner, such as certain rule-acquisition techniques, might be extracted as separate components, and run, real-time, alongside TextLearner rather than inside it, thereby reducing the work that TextLearner needs to do and consequently increasing its speed. (This has, in fact, been done already with the noun compound rule-learning technology.) The prospects for making a version of TextLearner that can run faster and more efficiently are, we believe, good.

[20] http://en.wikipedia.org/wiki/Chemistry

7 Significant Development

In previous sections of this report, a number of significant developments have been identified:

- a Method for representing the structure of text and html documents
- improved KB-driven methods for speeding up forward inference for applications
- a method of producing semantically rich text annotation, in the form of assertions that associate sentence-, phrase-, and word-level string occurrences with their possible CycL interpretations and a score weighing those interpretations
- methods for augmenting the results of syntactic parsing
- methods of acquiring rules for semantic interpretation
- methods resolving some kinds of anaphora
- methods of extracting some kinds of presuppositions

An additional significant development of this work has not yet been discussed, primarily due to the fact that it is a method of learning valuable information that, though TextLearner benefits from it, is not run as part of TextLearner's text-processing routines. Specifically, the Reading Learning Comprehension project has resulted in a method for weighing, or valuing noun compound rules in terms of *semantic focus*. For our purposes, a rule is more or less focused semantically to the degree that the actual noun compounds it covers are actual meaningful, relatively frequently used noun compounds of the relevant natural language (in this case English). To measure this property, rules were "run forward" so as to generate actual noun compound instances from the rule, following this procedure:

1. Find all strings which Cyc recognizes as satisfying the modifier syntactic and semantic constraints for the rule.

   For example, if the rule responsible for generating "egg salad" requires that the modifier be a noun or phrase with a noun head that denotes any specialization of `EdibleStuff`, *the Cyc term for the collection of all edibles, we would generate strings such as "beef," "tomato" and "salmon."*

2. Find all strings which Cyc recognizes as satisfying the head syntactic and semantic constraints for the rule.

   For example, if the same "egg salad" rule requires the head to be a noun denoting any specialization of `FoodComposite`*—the Cyc term for the collection of all food "composites" (food dishes or items that require a combination of ingredients)—we would generate strings such as "stew," "soup" and "tofu."*

3. Randomly select M of these and pair them together as candidate compounds.

For example, "beef stew," "tomato soup," "salmon tofu."

4. Use the rule to parse these combinations into the semantics given by the rule.

   For example, parsing "beef stew" as `(SubcollectionOfWithRelationToTypeFn Stew ingredients Beef)`, *or the collection of instances of stew that have beef as an ingredient.*

5. Filter out candidates whose semantics are recognized as violating the argument and inter-argument constraints (Cyc's semantic meaningfulness constraints) for the semantic relation.

   For example, rejecting "salmon tofu" on the grounds that no instance of `VegetarianCuisine` , *of which* `Tofu` *is a specialization, can have any* `Meat` *(of which* `Salmon` *is a specialization) among its* `ingredients`.

With this list, we:

1. generate a list of N candidate compounds,
2. check Yahoo! for the number of hits for the compound string, modifier string, and head string,
3. store test data for a rule in a file hash table
4. access this test data in the KB via removal support for the CycL predicate `#$datedCandidateNCTestHasResults`.

For purposes of scoring the originating noun compound rule, if the number of hits for the modifier or head is too low, the compound is treated as an outlier, and its number of hits neither counts for nor against the score of the originating rule. However, if head and modifier hits meet or exceed a threshold, the number of hits for the compound is considered: All such numbers are added and divided by the number of remaining compounds for a "semantic focus" score. The scores are used by TextLearner in the disambiguation of noun compounds. Appendix A of this report gives a table of these rules with their scores.

This procedure is vulnerable to rewarding a rule incorrectly by virtue of failing to detect false positives. Specifically, the semantics of the rule is not being tested directly, so that a rule that meaningfully, but incorrectly parses a commonly-occurring noun compound will be rewarded by virtue of that compound receiving a high number of hits. Consider, for example, an "elephant ivory" rule that parses the compound into the subcollection of the head semantics that are "derived from" an instance of the modifier semantics. This rule will succeed in parsing "dog biscuit" as the collection of biscuits derived from dogs, and since "dog biscuit" receives many Yahoo! hits, the rule will be rewarded.

To guard against this problem, a method of finding such false positives has been implemented as part of the online "teach Cyc" web-game. The web-game submits for

confirmation or discomfirmation, statements, in English, to online users who wish to teach Cyc whether a putative fact that it has been considering is true or false. For example, knowing that a tractor is a type of wheeled vehicle, and that most wheeled vehicles it knows of have brakes, it formulates the hypothesis, "every tractor has some brake as a proper physical part" which a human user can validate or repudiate. In order to catch false positives for noun compound rules, it generates hypotheses to the effect that the semantics provided by a rule for a high-scoring noun compound are correct. For example the web game would submit "a dog biscuit is a biscuit derived from dogs" for confirmation or discomfirmation. Consensus discomfirmation causes a forward rule to fire and record "dog biscuit(s)" as an counter-example for the rule. TextLearner looks for such counter-example "tags" and thus ignores that rules precision score, effectively assigning it a zero. Consensus confirmation, given the high-hit count of the noun compound in question, is treated as evidence that the compound deserves lexicalization: The semantic interpretation generated by the rule is reified (represented explicitly as a first-order term in the Cyc knowledge base) and associated with the compound in the Cyc lexicon.

8 Implications for Further Research

The ability to use the "teach Cyc" web-game to identify counter-examples to noun compound rules suggests the promise of using human consensus to validate or repudiate other pieces of knowledge that TextLearner acquires, reducing the need for manual review by an ontologist. For example, the Noun Learner- and WordNet-based methods for acquiring definitions of new concepts frequently generate conflicting results. However, since TextLearner keeps a record of the context where these new concepts were encountered, there is enough information to generate a meaningful "multiple choice"-style question asking which meaning (if any) of a word as it occurs in a particular sentence (or paragraph) is correct. A worthwhile research project would be to test the effectiveness of this method on not only noun compound rules and hypothesized definitions, but on any knowledge acquisition method, such as learning semantic translation templates, currently used in TextLearner.

Another promising area for future machine learning research is to determine the degree to which TextLearner can refine its existing disambiguation strategies on its own, or even learn new strategies. Many of the disambiguation strategies currently implemented in TextLearner involve rewarding pairs of interpretations of different expression pegs when those interpretations share a nearest `isa` to a collection of a specified type. For example, if a sentence or paragraph contain both the words "gold" and "oil," which in Cyc are ambiguous between `GoldColor` and `Gold` in the first case and `CrudeOil` and `VegetableOil` in the second, will reward `Gold` and `CrudeOil` on the grounds that they share a nearest `isa` of `NaturalResourceType`—which is distinguished from other collections that any of the competing interpretations might be nearest instances of due to its being an instance of `ClarifyingCollectionType`, a collection (specifically, the collection of all collections that are salient for distinguishing concepts) that is "targeted"

by a specific TextLearner disambiguation strategy. A worthwhile experiment would be to test whether Cyc could *discover* collections, such as `ClarifyingCollectionType` and others not already identified by ontologists, that are useful for disambiguation.

Bibliography of Published Works Sponsored by this Project

Curtis, Jon, D. Baxter, J. Cabral, On the Application of the Cyc Ontology to Word Sense Disambiguation. In *Proceedings of the Ninetheenth International FLAIRS Conference.* pp. 652-657, Melbourne Beach, Florida May 11-13, 2006.

Appendix A

Below are the results of a query that retrieves the semantic focus scores of Cyc's existing noun compound rules. Note that rules with a compositional syntax formed from `NCRuleWithTemplateFn` are derived from TextLearner's rules for generating noun compound rules in response to encountering unfamiliar compounds.

```
Mt : (MtSpace CurrentWorldDataCollectorMt-NonHomocentric (MtTimeDimFn
Now))
EL Query :
(thereExists ?RESULT
   (and
       (aggregateCNCResultForRuleWithThresholds ?RULE 10 1000 ?RESULT
Yahoo-SearchEngine)
       (evaluate ?SCORE
           (NthInListFn ?RESULT 2))))
```

Status : Suspended, Exhaust Total

Show Hide **Answers** (127 new) **Actions :** [Save Answers] [Graph Answers]

?RULE	?SCORE
MilkTruck-NCR	0.04954954954954955
DLAForwardLAN-NCR	0.052
(RelationalNCRuleFn costars 2 (WordWithPrefixFn Co_Denominal-ThePrefix Star-TheWord))	0.04
(RelationalNCRuleFn properParts 2 Section-TheWord)	0.4362934362934363
(NCRuleWithTemplateFn (TheList SubcollectionOfWithRelationFromFn (TheList TheNCHead genls IDString) identificationStrings (TheList TheNCModifier isa Thing)))	0.10326086956521739
HarpDepot-NCR	0.13333333333333333
ShoeFactory-NCR	0.20603015075376885
(RelationalNCRuleFn-InstanceLevel altitudeAboveGround 2 Altitude-TheWord)	0.04819277108433735
(RelationalNCRuleFn guestAtSocialGathering 2 Guest-TheWord)	0.7423312883435583
(RelationalNCRuleFn-InstanceLevel recruiterForOrg 2 Recruit-TheWord)	0.06319702602230483
(RelationalNCRuleFn annualPercentageRate 2 APR-TheWord)	0.31343283582089554
(RelationalNCRuleFn subsetOf 2 Superset-TheWord)	0.03137254901960784
(RelationalNCRuleFn temperatureOfObject 2 Temperature-TheWord)	0.25547445255474455

AirlineExecutive-NCR	0.1660377358490566
TicketSales-NCR	0.12432432432432433
(RelationalNCRuleFn-InstanceLevel properParts 2 Section-TheWord)	0.25
(RelationalNCRuleFn-InstanceLevel eventPlannedBy 2 Plan-TheWord)	TheEmptyList
(RelationalNCRuleFn (PresentTenseVersionFn transportees) 2 Transportee-TheWord)	TheEmptyList
SkiingAbility-NCR	0.1917808219178082
BusTerminal-NCR	0.05855855855855856
ElephantIvory2-NCR	0.12598425196850394
(RelationalNCRuleFn-InstanceLevel subsetOf 2 Superset-TheWord)	0.02527075812274368
InfectedAnimalS-NCR	0.06093189964157706
ElectionDay-NCR	0.15217391304347827
(RelationalNCRuleFn chosenItem 2 Pick-TheWord)	0.6666666666666666
(NCRuleWithTemplateFn (TheList SubcollectionOfWithRelationToFn (TheList TheNCHead genls Situation) situationConstituents (TheList TheNCModifier isa Individual)))	0.07575757575757576
GunLaw-NCR	0.06607929515418502
CycorpStaffMember-NCR	0.0390625
(NCRuleWithTemplateFn (TheList SubcollectionOfWithRelationToTypeFn (TheList TheNCHead genls Situation-Localized) situationLocation (TheList TheNCModifier genls EnduringThing-Localized)))	0.13908872901678657
(NCRuleWithTemplateFn (TheList SubcollectionOfWithRelationToTypeFn (TheList TheNCHead genls Situation-Localized) situationConstituents (TheList TheNCModifier genls SpatialThing-Localized)))	0.16951788491446346
PartyGame-NCR	0.096
BasketballEquipment-NCR	0.1910569105691057
(RelationalNCRuleFn layers 2 Layer-TheWord)	TheEmptyList
FleetSupport-NCR	0.6666666666666666
(RelationalNCRuleFn stepfather 2 (WordWithPrefixFn Step-ThePrefix Father-TheWord))	0.02491103202846975

(RelationalNCRuleFn isLeaderOf 1 Leader-TheWord)	0.4714285714285714
TourEvents-NCR	0.6885813148788927
(RelationalNCRuleFn wasTheKillerOf 2 Kill-TheWord)	0.31835205992509363
AnthraxExposure-NCR	0.4956521739130435
(NCRuleWithTemplateFn (TheList SubcollectionOfWithRelationToTypeFn (TheList TheNCHead genls BilateralObject) objectSides (TheList TheNCModifier genls SpatialThing-Localized)))	0.13559322033898305
AnthraxBacillus-NCR	0.5
NutritionConsultant-NCR	0.5
(RelationalNCRuleFn-InstanceLevel goals 2 Goal-TheWord)	0.2131782945736434
(NCRuleWithTemplateFn (TheList SubcollectionOfWithRelationFromTypeFn (TheList TheNCHead genls IDString) identificationStrings (TheList TheNCModifier genls Thing)))	0.12735849056603774
CountrysideLifeseeing-NCR	0.23550724637681159
TelanganaRashtraSamitiLeader-NCRule	0.20437956204379562
(NCRuleWithTemplateFn (TheList SubcollectionOfWithRelationToTypeFn (TheList TheNCHead genls EventSeries) seriesMembers (TheList TheNCModifier genls Event)))	0.5588235294117647
(NCRuleWithTemplateFn (TheList SubcollectionOfWithRelationToFn (TheList TheNCHead genls LexicalWord) wordInLanguage (TheList TheNCModifier isa NaturalLanguage)))	0.05247376311844078
TourLandmarkS-NCR	1
(RelationalNCRuleFn dividendPerShare 2 Dividend-TheWord)	1
(NCRuleWithTemplateFn (TheList SubcollectionOfWithRelationToTypeFn (TheList TheNCHead genls GeneralizedTransfer) terminalOfTrans-Generic (TheList TheNCModifier genls Individual)))	0.17727272727272728
(RelationalNCRuleFn boardMembers 2 Director-TheWord)	0.4681647940074906
(RelationalNCRuleFn beneficiary 2 Benefit-TheWord)	0.17454545454545456
PumpkinPatch-NCR	0.2593856655290102
RiverRafting-NCR	0.21299638989169675
(RelationalNCRuleFn-InstanceLevel teacherOf 2 Teach-TheWord)	0.3333333333333333
BloodSample-NCR	0.32081911262798635

RailTravel-NCR	0.30837004405286345
(RelationalNCRuleFn mySentientUserProfile 2 Profile-TheWord)	0.398576512455516
(RelationalNCRuleFn-InstanceLevel isLedBy 2 Leader-TheWord)	0.14691943127962084
(NCRuleWithTemplateFn (TheList SubcollectionOfWithRelationToTypeFn (TheList TheNCHead genls LinkingTogether) linkedThing (TheList TheNCModifier genls Thing)))	0.26666666666666666
AnthraxOutbreak-NCR	0.16806722689075632
(RelationalNCRuleFn-InstanceLevel claims 2 Claim-TheWord)	0.268
LogisticsCompany-NCR	0.09375
(RelationalNCRuleFn isLeaderOf 2 Follow-TheWord)	0.16363636363636364
(NCRuleWithTemplateFn (TheList SubcollectionOfWithRelationToTypeFn (TheList TheNCHead genls Situation) situationConstituents (TheList TheNCModifier genls Individual)))	0.12442396313364056
PoliceVehicle-NCR	0.11885245901639344
PollingStation-NCR	0.21739130434782608
FosterDaughter-NCR	0.5
(RelationalNCRuleFn halfBrothers 2 Half-Brother-MWW)	0.04460966542750929
LandscapeContractor-NCR	0.2078853046594982
(RelationalNCRuleFn denies 2 Deny-TheWord)	0.16370106761565836
(NCRuleWithTemplateFn (TheList SubcollectionOfWithRelationFromTypeFn (TheList TheNCHead genls (SubcollectionOfWithRelationFromTypeFn Individual parts Individual)) parts (TheList TheNCModifier genls Individual)))	0.140625
SoftwareEncryption-NCR	0.17391304347826086
(RelationalNCRuleFn employees 1 Employ-TheWord)	0.23674911660777384
(RelationalNCRuleFn-InstanceLevel subsetOf 1 Subset-TheWord)	0.07063197026022305
(RelationalNCRuleFn intelligenceOperativeForOrg 2 Spy-TheWord)	0.11026615969581749
TofuStew-NCR	0.3586206896551724
WaterSolution-NCR	0.44594594594594594
(NCRuleWithTemplateFn (TheList SubcollectionOfWithRelationToTypeFn (TheList TheNCHead genls Series) seriesMembers	0.09734513274336283

(TheList TheNCModifier genls TemporalThing)))	
ChristianSchool-NCR	0.32867132867132864
(RelationalNCRuleFn income 2 Earnings-TheWord)	0.1795774647887324
ChainSawPrices-NCR	0.6886446886446886
DriedYams-NCR	0.017605633802816902
CountryBoy-NCR	0.475
(RelationalNCRuleFn-InstanceLevel temperatureOfObject 2 Temperature-TheWord)	0.11023622047244094
(RelationalNCRuleFn-InstanceLevel mySentientUserProfile 2 Profile-TheWord)	0.2648221343873518
LeatherPants-NCR	0.12
BookCopies-NCR	0.5609756097560976
(RelationalNCRuleFn-InstanceLevel isLeaderOf 1 Leader-TheWord)	0.10572687224669604
(RelationalNCRuleFn rumorAbout 2 Rumor-TheWord)	0.21660649819494585
USSenateTestimony-NCR	0.4016393442622951
PassportPhotograph-NCR	0.245136186770428
(RelationalNCRuleFn (PresentTenseVersionFn transporter) 2 Transport-TheWord)	0.11219512195121951
AnthraxVaccination-NCR	0.24519230769230768
(NCRuleWithTemplateFn (TheList SubcollectionOfWithRelationToTypeFn (TheList TheNCHead genls Thinking) objectOfMentalSituation (TheList TheNCModifier genls SomethingExisting)))	0.19736842105263158
HomePhoneNumber-NCR	0.5178571428571429
USArmyMajors-NCR	0.06439393939393939
RenaissanceArtwork-NCR	0.3076923076923077
(RelationalNCRuleFn associates 1 Associate-TheWord)	0.24731182795698925
SkinContact-NCR	0.2627986348122867
(RelationalNCRuleFn intelligenceOperativeForOrg 2 Operative-TheWord)	0.21897810218978103
(RelationalNCRuleFn-InstanceLevel cost 2 Cost-TheWord)	0.14112903225806453
ElephantIvory1-NCR	0.04975124378109453
StarbucksFranchise-NCR	0.043478260869565216
DefenseDepartmentFacility-NCR	0.10861423220973783
ChicagoBusinessman-NCR	0.04201680672268908

(NCRuleWithTemplateFn (TheList SubcollectionOfWithRelationToFn (TheList TheNCHead genls PartiallyIntangibleIndividual) intangibleParts (TheList TheNCModifier isa IntangibleIndividual)))	0.06818181818181818
(RelationalNCRuleFn supervisesInAction 1 Supervise-TheWord)	0.3107142857142857
CourtRuling-NCR	0.10227272727272728
(RelationalNCRuleFn-InstanceLevel rivals 2 Rival-TheWord)	0.09523809523809523
SkinRash-NCR	0.3278008298755187
(RelationalNCRuleFn shareholders 2 Stockholder-TheWord)	0.25
(RelationalNCRuleFn suppliers 2 Supply-TheWord)	0.2426470588235294
PublicBus-NCR	0.6218181818181818
CanadianNavyManual-NCR	0.07851239669421488
(RelationalNCRuleFn subsetOf 1 Subset-TheWord)	0.24124513618677043
(RelationalNCRuleFn-InstanceLevel (InverseBinaryPredicateFn arrests) 1 Arrestee-TheWord)	TheEmptyList
SkinCondition-NCR	0.13043478260869565
(RelationalNCRuleFn downtownOf 2 Downtown-TheWord)	0.5
(NCRuleWithTemplateFn (TheList SubcollectionOfWithRelationToTypeFn (TheList TheNCHead genls PhysicalSituation) situationConstituents (TheList TheNCModifier genls PartiallyTangible)))	0.13106796116504854
(RelationalNCRuleFn accomplices 2 Accomplice-TheWord)	0.07037037037037037
(RelationalNCRuleFn chosenItem 2 Select-TheWord)	0.6538461538461539
ExposedMonkeys-NCR	0.04678362573099415
BathroomEquipment-NCR	0.5061728395061729
IRAMembers-NCR	0.2765151515151515
MapleLeaf-NCR	0.091324200913242